RADC-TDR-64-149, Vol III
Final Report

ERROR CONTROL THROUGH CODING

Volume III - Variable Redundancy Codes

TECHNICAL DOCUMENTARY REPORT NO. RADC-TDR-64-149

July 1964

Communications Research Branch
Rome Air Development Center
Research and Technology Division
Air Force Systems Command
Griffiss Air Force Base, New York

760 C System
Project No. 4519, Task No. 451903

(Prepared under Contract AF30(602)-2958 by R.T. Chien and D. T. Tang, International Business Machines Corporation, Federal Systems Division, Bethesda, Md.)

TABLE OF CONTENTS

AIR FORCE, GOODWAY PRINTING CO., PHILADELPHIA, PA., 9/28/64, 146

ERROR CONTROL THROUGH CODING

VOLUME III. VARIABLE REDUNDANCY CODES

The main results of the research work on Variable Redundancy Codes fall into the following categories:

1. Classes of variable redundancy codes.

2. Implementation of variable redundancy codes.

3. Mode control of variable redundancy codes.

While special efforts were undertaken to investigate each of the above categories in detail, the close relationships and interactions of the solutions to these problems were inevitable. An ideal variable redundancy scheme must combine efficient codes, simple implementation, and reliable mode control to form a coherent system.

Before summarizing the results in each of the three categories, we shall first introduce some basic terminology, notations, and conventions used throughout the work reported here.

The generator polynomial of a cyclic (n, k) code C is denoted by g(x) of degree r, where $r = n - k$. The natural code length n is the period of g(x), the smallest integer, such that $g(x) \mid x^n + 1$. The code C_h generated by $h(x) = \frac{x^n + 1}{g(x)}$ is the dual code of C. A code polynomial is a multiple of g(x) with degree $< n$. A code word or a code sequence corresponds to the coefficients of the code polynomial in descending order.

A single shift-register stage is characterized by a unit delay between its input and output terminals. The input-output relationship of a shift-register circuit can be described by a transfer function in terms of the delay operator D. Because of the "high-order-first" convention, D can be replaced by x^{-1}.

CLASSES OF VARIABLE REDUNDANCY CODES

Three classes of variable redundancy codes were developed:

1. DUAL CODES (REFERENCE: APPENDIX B)

The use of dual codes always gives two modes of operation. In one mode the code generator is g(x) and in the other mode the code generator is $h(x) = (x^n + 1) / g(x)$.

Because of the restriction $g(x)\, h(x) = x^n + 1$, it is often not possible to find g(x) such that both of the dual codes are optimum.

However, a number of efficient codes were found in each of the following combinations:

A. g(x) and h(x) both generate burst-error-correcting codes: Reasonably good codes can be obtained by splitting the factors of $x^n + 1$ into two disjoint parts.

B. g(x) and h(x) both generate independent-error-correcting codes: A special case is when g(x) is a primitive irreducible polynomial of degree m. C_g is a single-error-correcting code of length $= 2^m - 1$. The dual Code C_h, on the other hand, is a $(2^{m-2} - 1)$-error-correcting, 2^{m-2}-error-detecting code.

C. g(x) generates a burst-error-correcting code while h(x) generates an independent-error-correcting code: A special case is when $g(x) = (x + 1)\, p(x)$, where p(x) is a primitive irreducible polynomial of degree m. C_g is a burst-2 correcting code of length $= 2^m - 1$. The dual code C_h is a $(2^{m-2} - 1)$-error-correcting code.

The main limitation to the use of dual codes as variable redundancy codes is that the code is often too powerful in the high-redundancy mode. However, because of the fact that little extra circuitry due to the additional mode of operation is needed in decoding (this will be clear in the next section on implementation), this class of codes remains as a distinctive possibility.

2. PRODUCT CODES (REFERENCE: APPENDIX C)

The use of product codes gives two or more modes of operation. Let g(x) be the generator polynomial of code C_1 used in one mode; then the generator polynomial of code C_2 used in the other mode takes the form of $g_1(x)\ g_2(x)$. Another factor may be added to form the generator of an even more powerful code.

Many efficient product codes exist in each of the following combinations:

A. Burst-error-correcting product codes: A number of optimum burst-error-correcting product codes can be constructed according to the following theorem:

Theorem: Let $g(x) = g_1(x)\ \ g_2(x)$ be the generator polynomial of a burst-b correcting code, and let r_2 be the degree of $g_2(x)$. Then $g_1(x)$ generates a burst-b_1 correcting code, where $b_1 \geq b - r_2$.

It is shown that, if the code generated by g(x) is optimum, the code generated by $g_1(x)$ is also optimum, provided that $g_1(x)$ has the same period as g(x).

B. Independent-error-correcting product codes: The class of Bose-Chaudhuri-Hocquenghem codes can be immediately seen to be in the exact form of product codes. For if $g_1(x)$ is the minimal polynomial containing α, α^2, ..., α^{2t_1} as roots, where α is a primitive element in GF (2^m), then $g_1(x)$ generates a t_1-error-correcting code. The code generated by g(x), the minimal polynomial containing α, α^2, ..., α^{2t} as roots, where $t > t_1$, is a t-error-correcting code satisfying $g(x) = g_1(x)\ \ g_2(x)$.

C. Multiple-burst-correcting product codes: The general BCH code over GF (q^m) where $q \neq 2$ can be used as product codes correcting multiple bursts. The class of Reed-Solomon codes is a special case.

The class of product codes is believed to be the most promising among the various classes of variable redundancy codes. Its major attraction lies in the fact that efficient codes can be obtained over a wide range of redundancy specifications.

3. VARIABLE LENGTH CODES: (REFERENCE: APPENDIX D)

Apart from the dual codes or product codes which have different natural lengths, a shortened code can be regarded as a high-redundancy code in comparison with the original code. The error-correcting ability of shortened codes is often better than the original code up to a certain percentage of the original length. The set of optimum burst-3 and burst-4 codes obtained by Elspas were tested and some interesting shortened lengths with improved maximum bursts are tabulated.

IMPLEMENTATION OF VARIABLE REDUNDANCY CODES

The main achievements in the study of the implementation of variable redundancy codes include the analysis of basic building blocks and the development of an improved decoder which consists of a minimum of n shift-register stages and takes a minimum of n bits delay for decoding.

1. BASIC BUILDING BLOCKS (REFERENCE: APPENDICES A AND C)

A basic building block corresponding to g(x) is a shift register circuit defined by the transfer function in the delay operator D,

$$\frac{t}{s} = g^*(D) + 1,$$

where t is the output sequence, s is the input sequence, and $g^*(D) = D^r \cdot g(D^{-1})$ is the reciprocal polynomial of g(D). The addition here is binary. Such a building block can be connected to perform multiplication (Figure 1) or division (Figure 2), and is therefore essential to the shift-register circuits in general.

It is possible to combine building blocks corresponding to $g_1(x)$ and $g_2(x)$ to form a building block corresponding to $g(x) = g_1(x) \cdot g_2(x)$. This is an important concept concerning the implementation of product codes. Figure 3 shows such a block connected as a division circuit. There are two desirable features in this circuit:

A. The block is defined by the transfer function

$$\frac{t}{s} = g^*(D) + 1 = g_1^*(D) \cdot g_2^*(D) + 1 .$$

Therefore, by breaking the input at A, we have $s_2 = t_2 = 0$, and hence

$$\left(\frac{t}{s}\right)' = \frac{t_1}{s_1} .$$

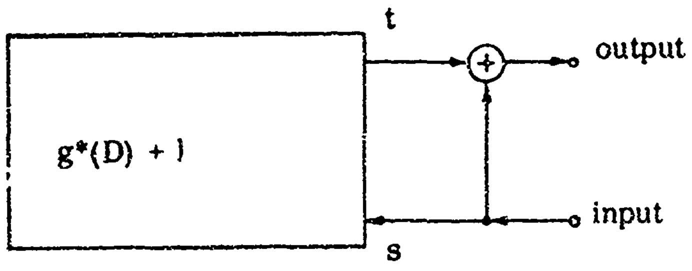


Figure 1

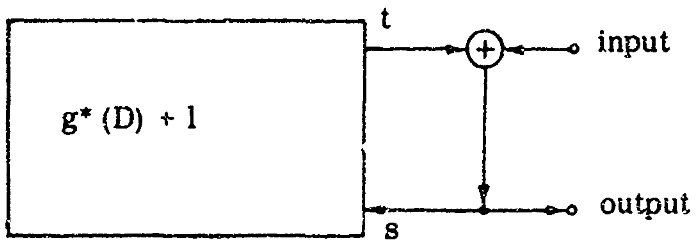


Figure 2

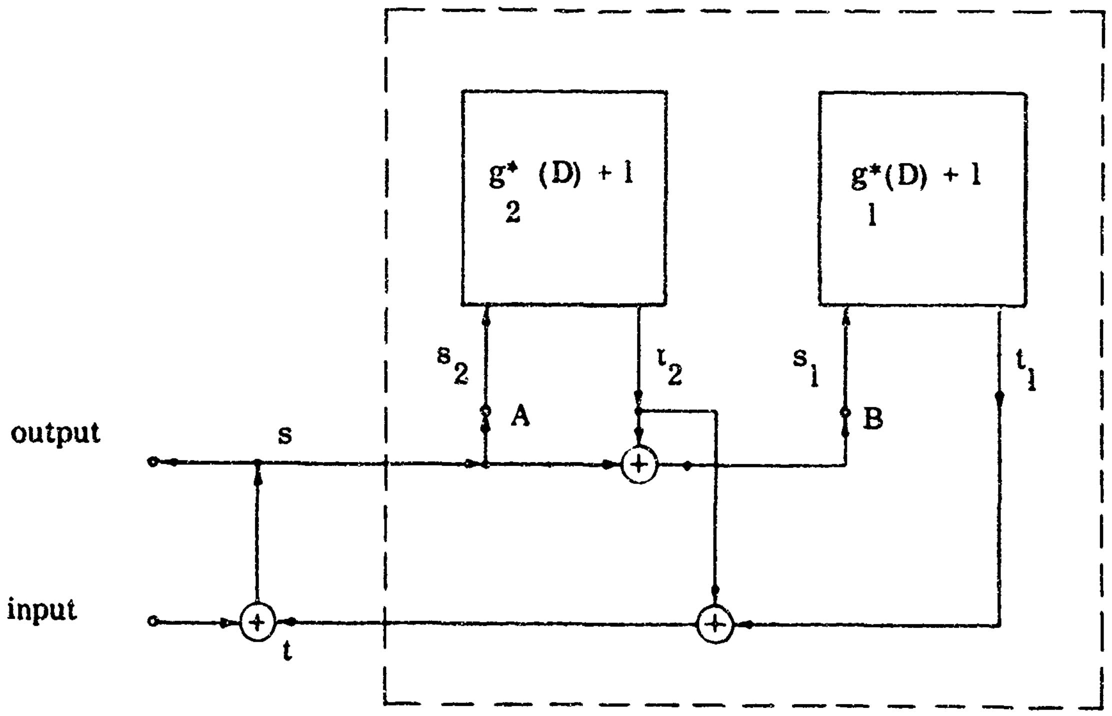


Figure 3

If we break the input at B, we have $s_1 = t_1 = 0$, and hence

$$\left(\frac{t}{s}\right)'' = \frac{t_2}{s_2}.$$

B. When the block is connected as a division circuit, we have

$$s + t + m = 0.$$

where m is the input sequence. It follows that,

$$\begin{aligned} s_1 + t_1 + m &= (s + t_2) + (t + t_2) + m \\ &= s + t + m \\ &= 0. \end{aligned}$$

This means that the block corresponding to $g_1(x)$ behaves as if it were connected as a division circuit by itself.

With the received sequence fed to the circuit, the shift-register contents of both blocks together can be regarded as the error syndrome corresponding to the code with g(x) as the generator. Simultaneously, the shift-register contents of the block on the right can be regarded as the error syndrome corresponding to the code with $g_1(x)$ as the generator.

Similar features can be obtained in the cases where three or more blocks corresponding to different factors of g(x) are combined. The versatility of the combined circuit is ideal in the implementation and mode control of product codes.

2. AN IMPROVED DECODER (APPENDICES A, B, AND C)

As can be seen in Figure 4, this improved decoder contains two building blocks corresponding to g(x) and h(x) respectively. The total number of shift-register stages is n. These building blocks are connected in such a way that the exact received sequence of n bits can be reproduced at the output after a delay of n bits. In addition, a syndrome recognizing circuit is added which provides the error correcting ability. For burst-error-correcting codes, the recognizing circuit consists of an OR - gate and an AND - gate.

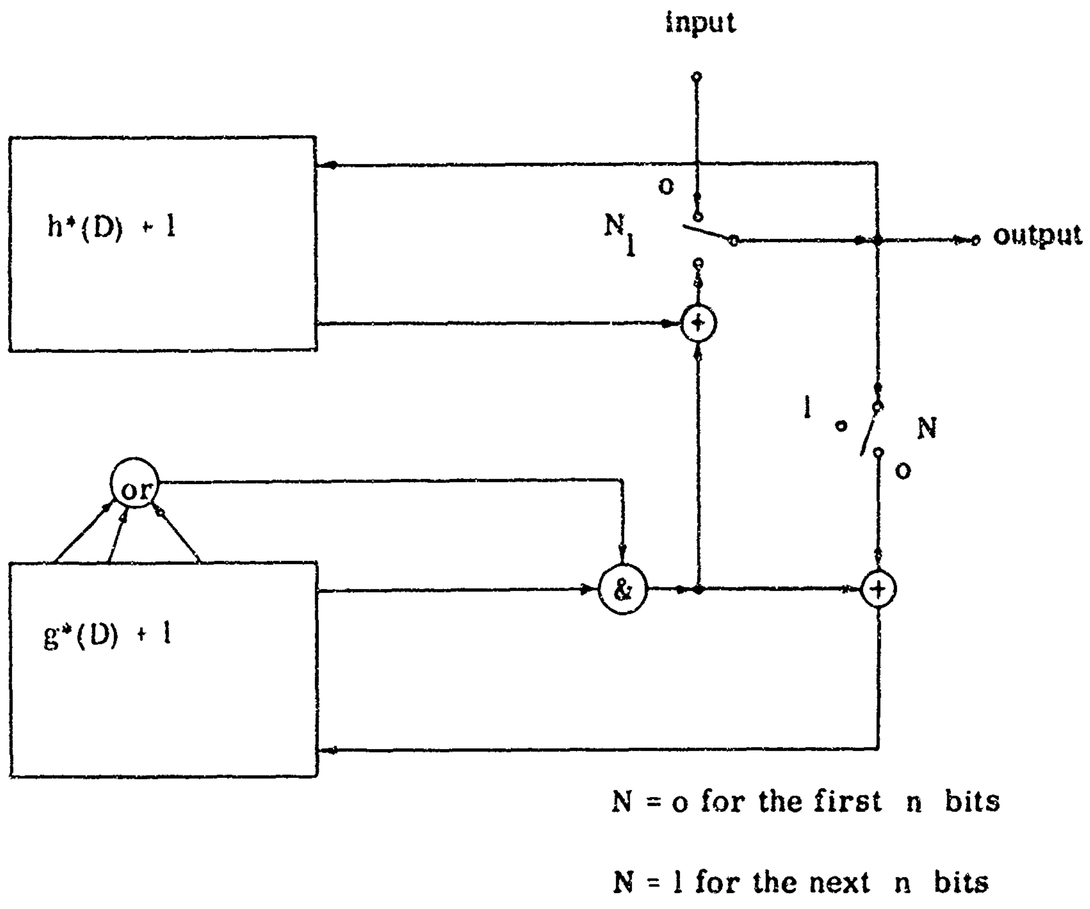
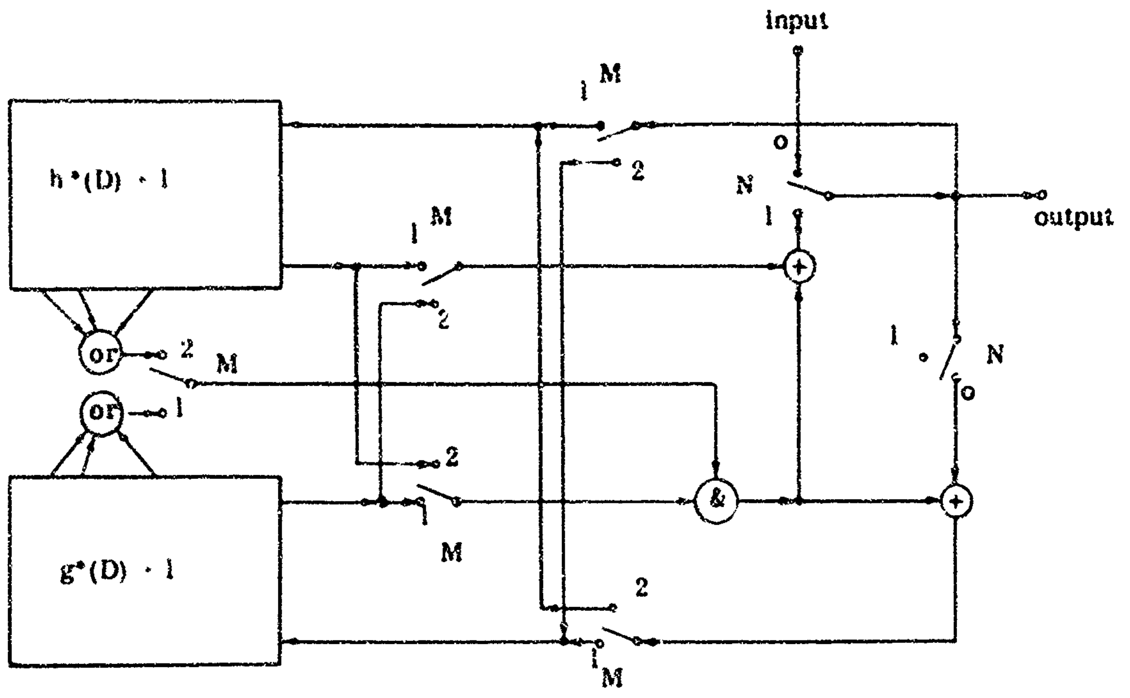


Figure 4

This type of encoder is ideal for dual codes. For it is only necessary to interchange the input and output connections of the two building blocks. Figure 5 shows such a realization.

For product codes, three or more blocks can be combined in this type of decoder. With $g(x) = g_1(x)\ g_2(x)$, the block corresponding to $g_2(x)$ is combined with the block corresponding to $h(x)$ in the low-redundancy mode to form a block corresponding to $h_1(x) = \dfrac{x^n + 1}{g_1(x)}$, and is combined with the block corresponding to $g_1(x)$ in the high-redundancy mode to form a block corresponding to $g(x)$.



M= 1 when g(x) is the generator

M = 2 when h(x) is the generator

Figure 5

MODE CONTROL OF VARIABLE REDUNDANCY CODES

Five approaches were investigated in the mode control of variable redundancy codes. They are all included in Appendix F.

1. EACH MESSAGE BLOCK CONTAINS A MODE INDICATION

It is shown that when a single bit is used to indicate the mode of the next block, the protection of this bit is much stronger than an ordinary code polynomial if the percentage redundancy is low. Thus, the bit always gets strong protection in all modes of operation. This approach can be used for all classes of variable redundancy codes, and is particularly suitable if the change of mode occurs quite frequently.

2. EVERY W'TH MESSAGE BLOCK CONTAINS THE MODE INFORMATION

Extra hardware necessary to transform the mode information into actual control may be expensive. The delay of w blocks in some cases may also be undesirable.

3. USE AN INTERRUPT SEQUENCE FOR THE INDICATION OF MODE CHANGES

As a special case, such an interrupt sequence may consist of two parts: The first part serves to identify the sequence (a run of ones for instance), while the second part serves to establish synchronization with error protection (a maximum length sequence for instance). This approach is independent of the classes of codes used, and is appropriate if the change of mode does not occur very frequently, particularly so if the code length is large.

4. USE A SPECIAL CODE WORD FOR THE INDICATION OF MODE CHANGES

Since the special code word must occupy the exact length of a block, there is no synchronization problem in this case. Protection is obtained from the code itself. This approach can be used for all classes of codes. It's appropriate if the change of mode does not occur very frequently, and if the delay of n bits is acceptable.

5. NON-UNIFORM PARTITION OF THE BINARY SPACE

Two examples of different schemes were obtained with decoders completely worked out. Both are product codes with non-uniform partitions in the binary space each of which is to be mapped into a unique code sequence. Different degrees of error protection are thereby obtained.

The evaluation of this type depends heavily on the code efficiency and hardware complexity, and thus can only be investigated on an individual basis.

OTHER MISCELLANEOUS RESULTS

1. Some theorems of general interest were obtained. They are reported in Appendices B, C, and E.

Theorem: Let g(x) be the generator of C the cyclic code C of length n. Then $g^*(x)$ generates an equivalent code of the same length with the same maximum number of correctible burst or independent errors.

Theorem: Let α be any element of $GF(2^m)$ of order e. Then the code consisting of all vectors (g(x)) over the binary field for which

$$\alpha^{m_0}, \alpha^{m_0+s}, \alpha^{m_0+2s}, \ldots, \alpha^{m_0+(d-2)s}$$

are roots of f(x) is a Bose-Chaudhuri-Hocquenghem code with minimum distance at least d, provided $(s, e) = 1$.

Theorem: For any linear code, it is impossible to detect errors from a decoded message on which error-correction has been made.

2. Burst-Error-Correcting Codes generated from Multiplicative Cyclic Groups were Studied (Reference: Appendix G).

Syndromes of all correctible errors in this case must be in the multiplicative cyclic group containing polynomials relatively prime to the generator polynomial g(x) and with degree smaller than r. Let

$$g(x) = \prod_i g_i(x),$$

then the multiplicative cyclic group is obtained if the periods e_i of $g_i(x)$ are pairwise relatively prime.

It is shown that g(x) generates a burst-b-correcting code of this kind if and only if $b \leq r_i$ for all i, and the residues modulo g(x) of $B_i^e(x)$ are distinct. Where $B_i(x)$ ranges over all error patterns up to a burst of b, and $e = \prod_i e_i$ is the period of g(x).

Some new codes were found this way with a computer program. They are included in Appendix G.

APPENDICES

APPENDIX A

ON THE IMPLEMENTATION OF ERROR CORRECTING CODES

INTRODUCTION

The class of cyclic polynomial codes owes much of its popularity to the simplicity of its implementation. Shift-register circuits takes advantage of the cyclic structure of a code when properly connected and decodes with minimum delay. However, different approaches using different mathematical tools sometimes tend to confuse engineers who are trying to analyze or design the implementation of error-correcting codes. Such as the polynomial approach used by Peterson [7] and the matrix approach used by Meggitt [6]. Both investigate the transition of shift-register contents when analyzing the decoding process, although the syndromes are different.

It is shown in this report that, by analyzing the input-output relationships at certain key points in the circuit, fundamental requirements of an encoding or decoding circuit can be satisfied without directly looking at the shift-register contents. Many existing circuits can thus be seen to be equivalent, and some new ones can be added to satisfy specific needs.

This analysis also leads to two improved error-correcting circuits which are presented later in this report.

CYCLIC ERROR-CORRECTING CODES

Let $g(x)$ of degree $r = n - k$ be the generator polynomial of a cyclic (n, k) code C_g. A code word, represented in polynomial form, must be a multiple of $g(x)$. Also, $g(x) \mid x^n + 1$.

If the code C_g is capable of correcting a class of errors, $(E_i'(x))$, we must require each E_i' to be identified by a unique error syndrome which is either the residue of E_i' modulo $g(x)$ or some linear transformation of it which preserves the one-to-one correspondence between the errors and the syndromes.

For cyclic codes, it is convenient to represent all the cyclicly shifted versions of a particular error $E_i'(x)$ by a single $E_i(x)$, called an error pattern. The syndromes of all shifted versions of the same error pattern $E_i(x)$ fall in the same "cycle sets" of period n, unless there exists n' such that $x^{n'} E_i(x) \equiv E_i(x) \bmod (x^n + 1)$, in which case the period is the smallest such n' possible.

The error-correcting requirements of C_g can be written as follows:

For any two correctible error patterns $E_i(x)$ and $E_j(x)$ such that $E_i(x) \not\equiv x^m E_j(x) \bmod (x^n + 1)$, we have

$$x^k E_i(x) \not\equiv E_j(x) \qquad \bmod g(x) \tag{1}$$

or

$$x^k E_i(x)\, h(x) \not\equiv E_j(x)\, h(x) \bmod (x^n + 1) \tag{2}$$

for all $o \leq k \leq n - 1$, where h(x) is defined by

$$g(x)\, h(x) = x^n + 1 . \tag{3}$$

This alternative form of (2) is sometimes useful.

SOME EQUIVALENT CIRCUITS

In order to use various types of shift-register circuits for encoding and decoding, some equivalent circuits are first investigated. Here the equivalence is restricted to the input-output relationship only, and does not apply to the shift-register contents in general.

An important "building block" basic to an encoding or decoding circuit is one with the input-output relationship defined by the equation:

$$x^r \quad t = s \cdot g(x) + sx^r \tag{4}$$

where

$$g(x) = x^r + g_{r-1} x^{r-1} + \ldots + g_1 x + 1 \tag{5}$$

and s, t are sequences in the form of polynomials sent and received in the high-order-first manner.

For the sake of convenience (4) can be written in the form of a transfer function in x or in D:

$$\frac{t}{s} = \frac{g(x)}{x^r} + 1 = g^* (D) + 1, \tag{6}$$

where $D = x^{-1}$ is the delay operator and $g^*(x) = g(x^{-1})\, x^r$ is the reciprocal polynomial of g(x). Appropriate number of bits must be considered in s and t if (4) and (6) are interpreted as the polynomial of x. Abitrary number of bits in s and t are related by (4) and (6) when they are interpreted in the form of delay polynomials with $x^{-1} = D$.

Figure A-1 shows two circuits with the same transfer function in (6). The circuit of Figure A-1-A has mod 2 adders between shift-register stages while the circuit of Figure A-1-B has no adders between stages (thus can sometimes be replaced by tapped delay-lines). As far as the input-output relationship is concerned, these two circuits are interchangeable.

Figure A-2 shows the basic building block of Figure A-1 connected as a multiplication circuit, since if the input is $p(x)\, x^r$, we have

$$\text{output} = p(x)\, x^r \left(\frac{g(x)}{x^r} + 1 + 1 \right) = p(x)\, g(x) \tag{7}$$

Figure A-3 shows a similar building block connected as a division circuit, since if the input is p(x), we have s = output, and from (4),

$$p(x)\, x^r = (s + t)\, x^r = s \cdot g(x). \tag{8}$$

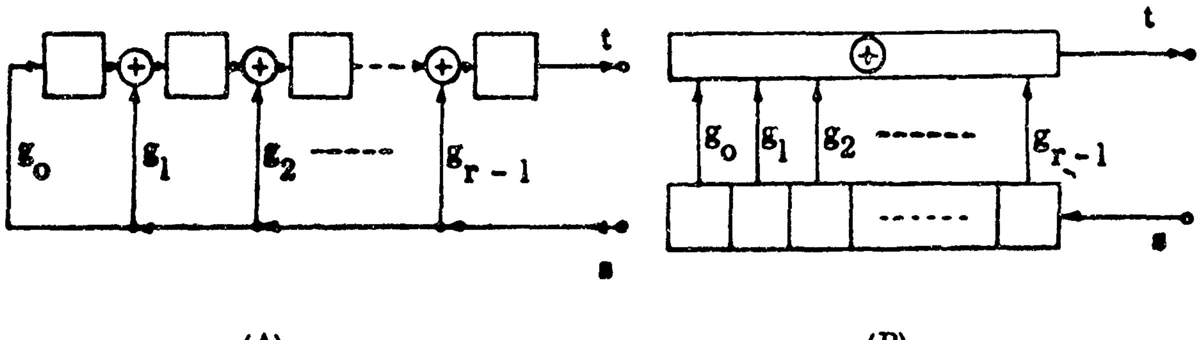


Figure A-1.

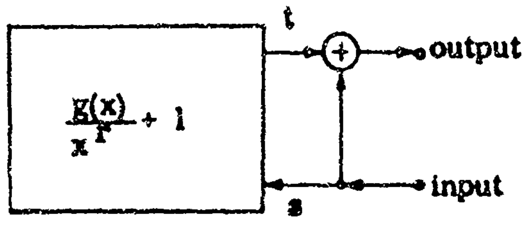


Figure A-2.

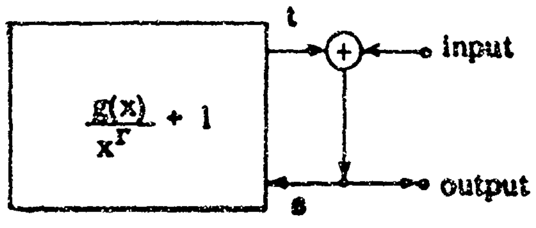


Figure A-3.

Therefore,

$$\text{output} = s = x^r\, p(x) / g(x) \tag{9}$$

Let $g(x) = g_1(x)\, g_2(x)$, where r_1 and r_2 are degrees of $g_1(x)$ and $g_2(x)$. Then it is possible to obtain the building block with transfer function $g(x)/x^r + 1$ from two building blocks with $\frac{g_1(x)}{x^{r_1}} + 1$ and $\frac{g_2(x)}{x^{r_2}} + 1$ as transfer functions. This is shown in Figure A-4.

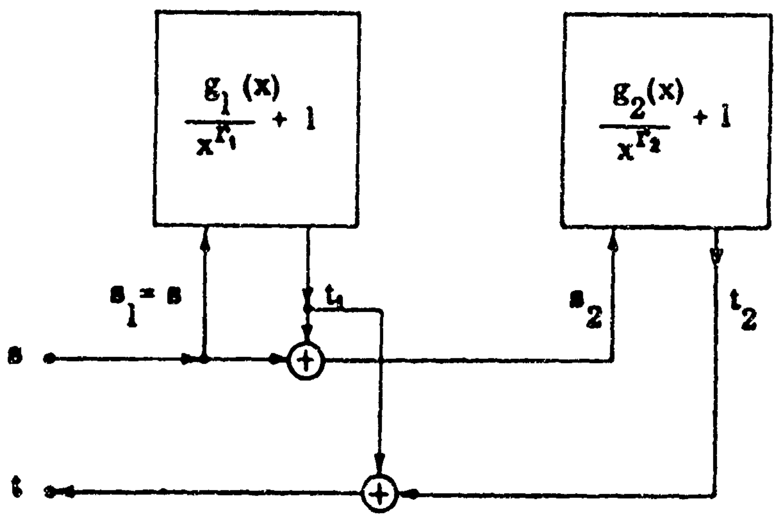


Figure A-4.

Here we have:

$$t = t_1 + t_2 = s\left(\frac{g_1(x)}{x^{r_1}} + 1\right) + s\,\frac{g_1(x)}{x^{r_1}}\left(\frac{g_2(x)}{x^{r_2}} + 1\right)$$

$$= s\left(\frac{g_1(x)\,g_2(x)}{x^{(r_1 + r_2)}} + 1\right) = s\left(\frac{g(x)}{x^r} + 1\right), \tag{10}$$

which is the result desired.

With proper switching, the circuit of Figure A-4 can be connected to perform multiplication or division corresponding to $g_1(x)$, $g_2(x)$, or $g(x) = g_1(x)\ g_2(x)$. The amount of complication introduced by putting two smaller building blocks together is small, and is also independent of $g_1(x)$ and $g_2(x)$, which is a very desirable feature.

CLOSE-LOOP SOLUTIONS

A close-loop solution of the circuits shown in Figure A-1 is a sequence existing when the input is directly connected to the output, i. e., when $s = t$. From (4)

$$x^r \cdot t = s \cdot g(x) + s \cdot x^r = x^r \cdot s \tag{11}$$

Since the period of all the roots of the characteristic function $g(x) = 0$ divides n, $s = t$ must have a period which divides n. (11) now becomes

$$s \cdot g(x) \equiv 0 \qquad \text{mod } (x^n + 1) \tag{12}$$

which means

$$s \equiv 0 \qquad \text{mod } h(x) \tag{13}$$

Therefore, the close-loop solution with $s = t$ must be a sequence of period n such that each complete cycle corresponds to a multiple of h(x). Conversely, if s has period = n, and for each cycle $s = 0$ mod h(x), then $s = t$ mod $(x^n + 1)$.

It is clear that r consecutive bits of the close-loop solution determine uniquely the solution. In particular, if we have $r - 1$ zeroes follcwed by a "1", then the first n bits of $s = t$ must correspond to $h(x)$, since $h(x)$ is the only non-zero polynomial which has degree no more than k, and is a multiple of $h(x)$. It is interesting to note that this particular close-loop solution can be obtained by feeding a sequence of single "1" (followed by zeroes) to a division circuit. The first n bits of $x^n/g(x)$ are apparently identical to that of $(x^n + 1)/g(x) = h(x)$. The remainder of "1" may be considered as the input of the next cycle and thus repeats with the same period of n.

The close-loop solution of a division circuit due to any input (followed by zeroes) can be obtained by adding the individual close-loop solutions resulting from the "1's" in the input.

ENCODING CIRCUITS

For separable codes, a k-bit message $m(x)$ is coded as:

$$C_m(x) = x^r m(x) + r(x) \tag{14}$$

where

$$r(x) \equiv x^r m(x) \qquad \text{mod } g(x) \tag{15}$$

Figure A-5 shows an encoding circuit for separable codes. Switch K is at "0" position for the first k bits when the information is loaded into the circuit. Check bits $r(x)$ is obtained after K is switched to the "1" position. This can be seen from the fact that with K at position "0", we have a division circuit. Thus if the input were $C_m(x)$, instead of $x^r m(x)$, we must have $s = 0$ for the last r bits since $g(x) \mid C_m(x)$. It follows that $t = C_m(x)$ for the last r bits. Therefore, by switching K to "1" after $x^r m(x)$ is loaded into the circuit, both s and t must be the same as if k is at position "0" and input $= C_m(x)$. In the meanwhile, $r(x)$ is obtained by taking the last r bits of t.

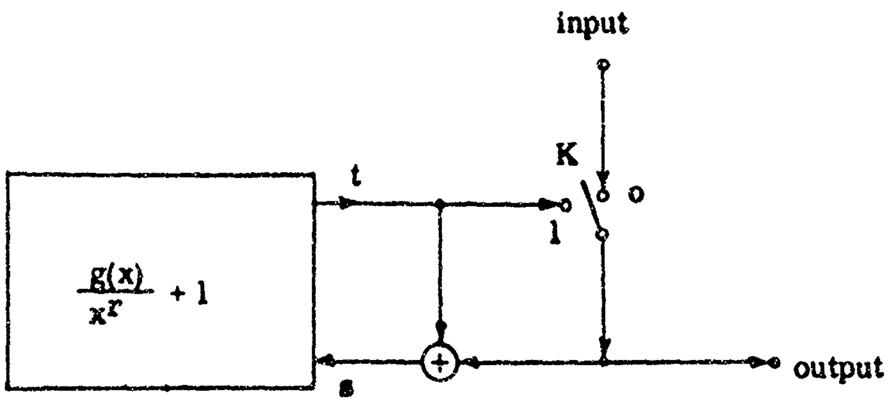


Figure A-5.

ERROR SYNDROMES AND ERROR PATTERNS

The close-loop solution of a division circuit depends on the input only. There are various equivalent circuits possible as long as (6) is satisfied. The question is: can we always regard the shift-register contents as error syndromes of the code C_g no matter which equivalent circuit is chosen?

The answer to the above question is affirmative. To show this, let's consider the cases where a minimum number of r shift-register stages are present. It can be seen that for any two distinct errors (either the same error pattern at different positions or different error patterns at all possible positions), the corresponding shift-register contents must be different. For if the shift-register contents are the same, the close-loop solution of period n must also be the same. This implies $E_1'(x)\ h(x) = E_2'(x)\ h(x)$ modulo $(x^n + 1)$, contradicting (2) if $E_1'(x)$ and $E_2'(x)$ are two distinct correctible errors.

Note that after n bits of the received sequence is shifted into the division circuit, the code word part (being a multiple of g(x)) leaves nothing in the shift-register, ana error syndromes go through an error cycle corresponding to a close-loop solution of the division circuit.

BURST ERROR SYNDROME DETECTION

Since it takes a complete received polynomial to determine the error syndrome, a minimum delay in decoding is n bits. It is therefore satisfactory if the syndrome is recognized n - 1 bits after the first bit or error enters the decoding circuit. Actually this often makes the syndrome detection circuit much simpler. In case of b burst-error correcting codes, n - 1 bits after the first bit of error enters the decoding circuit and at no other time, we have:

A. the next bit in s = t is a "1".

B. the contents of the first r - b shift-registers along the longest path from input to output are all "0's".

To demonstrate the above assertion, consider the close-loop solution corresponding to a single error x^i. The coefficients of h (x) are ready to appear after n - i - 1 shifts, and hence r - 1 zeroes have been observed in s = t at this time. It follows that the shift-registers contain r - 1 zeroes and a "1" in the last register at this time. Let's now look at any single error x^j alone, $1 \leq i - j \leq b - 1$. After n - 1 - i shifts, the corresponding s' = t' due to x^j has produced r - 1 - i + j zeroes. Thus the next bit in s' = t' must be a "0", and the first r - 1 - i + j shift-registers along the longest path from input to output must contain all zeroes. Since the circuit is linear, these syndromes are additive. The proof is completed by pointing out the one to one correspondence between syndromes and errors.

It is clear that the sequential decoding schemes used by Meggitt [6] and Peterson [7] are equivalent since they use the equivalent circuits of Figure A-1A and A-1B for the building blocks.

Figures A-6 and A-7 are two decoding circuits for the optimum burst-3 code generated by

$$g(x) = (x^2 + x + 1)\ (x^4 + x + 1) \qquad (16)$$

The function of these decoding circuits is clear from the analysis of this section. Note that the same logic circuit is used in both cases for the syndrome detection. Such circuits are suitable for multi-mode operation if this is one of the system's requirements.

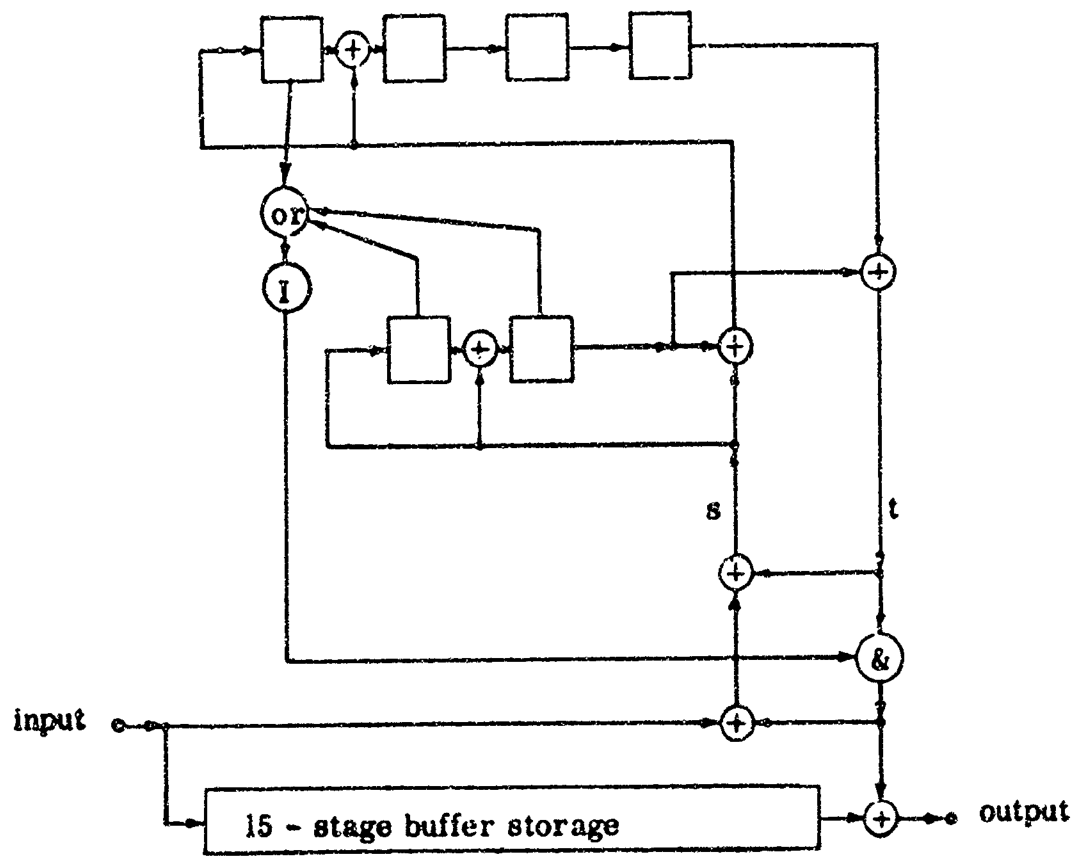


Figure A-6

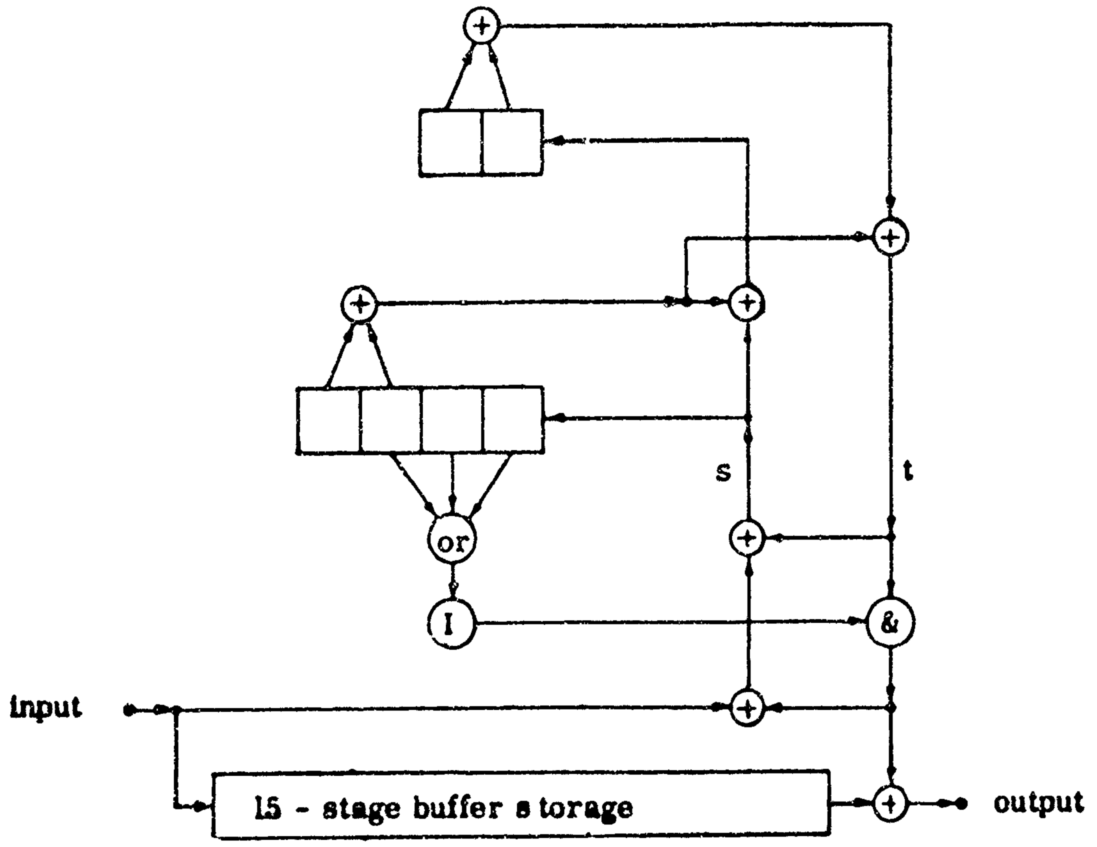


Figure A-7

It is possible to use a k - stage feed-back shift-register circuit instead of the n - stage buffer storage and thus reduce the total number of shift-register stages from n + r to n. The "error-trapping" approach as described by Rudolph and Mitchell [8] takes 2n bits of processing: These first n bits are the loading cycle. The next n bits are the decode cycle during which errors are kept out of the k - stage shift-register circuit. The original code word is obtained after a delay of 2n bits if the error is correctible.

We shall show that it is not necessary to keep the errors from the k - stage shift-register circuits as long as the received polynomial can be generated. The corrected code word can be obtained after a delay of n-bits. This will be demonstrated in the following in terms of the building blocks so that the use of equivalent circuits becomes immediate.

Figure A-8 shows a burst-error correcting circuit where g(x) is the generator polynomial of degree r and $h(x) = x^n + 1/g(x)$ is of degree k. Switches designated by N are at position "0" for the first n bits and at position "1" for

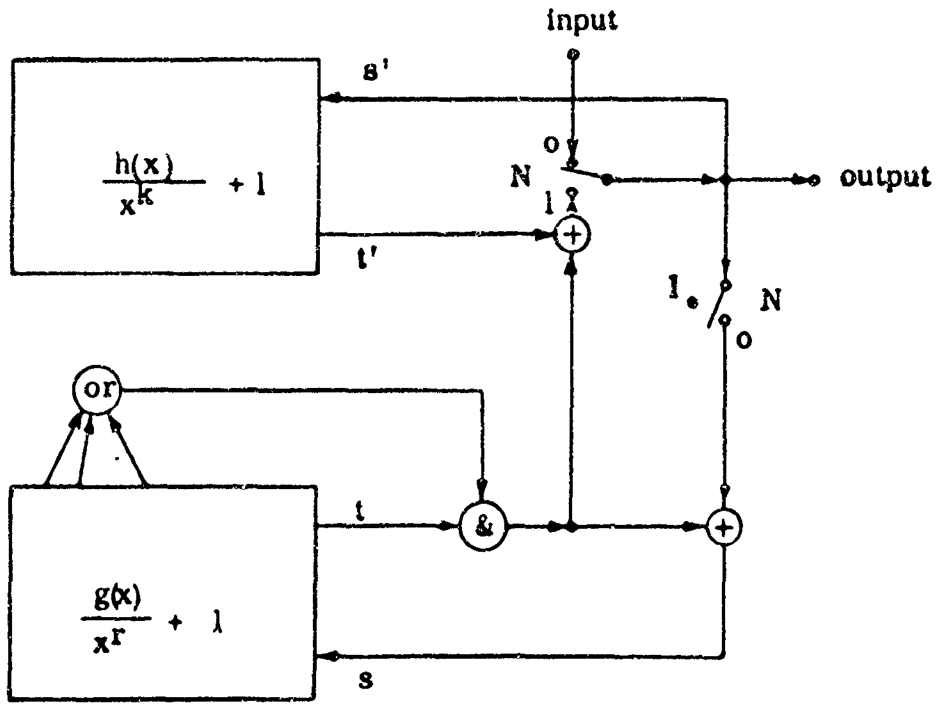


Figure A-8

the remaining n bits. The input is an n bit sequence followed by n zeroes and the output code word is obtained after n bits (the first n bits at the output are the received message).

Let $C'_m(x)$ be the sum of the original code polynomial $C_m(x)$ and an error polynomial $E(x)$. It is clear that, since the G-register is connected as a division circuit, $C_m(x)$ has no effect on the G-register contents at the end of n bits. After N is switched to position "1", the G-register contents correspond to syndromes going through their error cycles. The result of the last Section therefore applies: n-1 bits after the first error bit enters the decoding circuit, the syndrome can be recognized if the error is correctible.

Since the code word part $D_m(x)$ is the close-loop solution of the H-register, $C_m(x)$ will start reappearing as part of the output after n bits. In the meanwhile, after a single error enters the decoding circuit, the next $n - 1$ bits in both t and t' caused by the error bit are h_{k-1}, h_{k-2}, --- h_1, h_0, followed by $r - 1$ zeroes. Apparently they cancel each other at s' and the output. If no correction takes place, the next bit caused by the single error will be "0" at t' but will be "1" at t which means the following $n - 1$ bits again cancel each other at s' and the single error would reappear as part of the output after a delay of n bits. Since the circuit is linear, the effects of error bits on the output are additive. It follows that if the detection of $r - b$ zeroes fails (an uncorrectible error is detected), the received polynomial will reappear unaltered at the output. The correctible burst errors are corrected sequentially with the corresponding syndromes of each corrected error subtracted from the G-register contents.

Figure A-9 shows a decoding circuit similar to the one in Figure A-8. The difference is that, for the first n bits, the G-register is not closed to form a division circuit. However, the H-register is connected as a multiplication circuit. Thus, s, the input to the G-register due to a received polynomial in the first n-bit cycle $C'_m(x)$, is the first n bits of $C'_m(x)\,h(x)$. From Section 4, we see that this is the same as if the G-register is connected as a division circuit with $C'_m(x)$ as the input. Since the circuit is the same as the one in Figure A-8 during the second n-bit cycle, the overall performance is the same.

The circuit in Figure A-9 is somewhat similar to the "error trapping" type decoding circuit of Mitchell.

Using the general combinational circuit in place of the zero detector as the error syndrome recognizer, the decoding circuits of Figures A-8 and A-9 can be used for any cyclic code in principle. The circuit in Figure A-10 is a modified version of the circuit in Figure A-8.

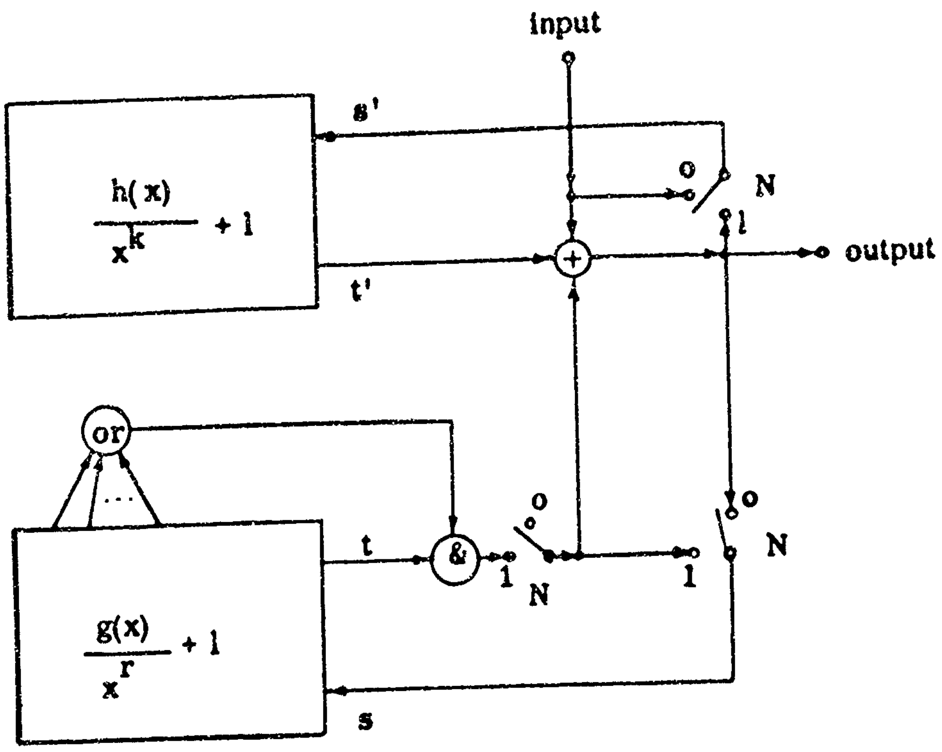


Figure A-9

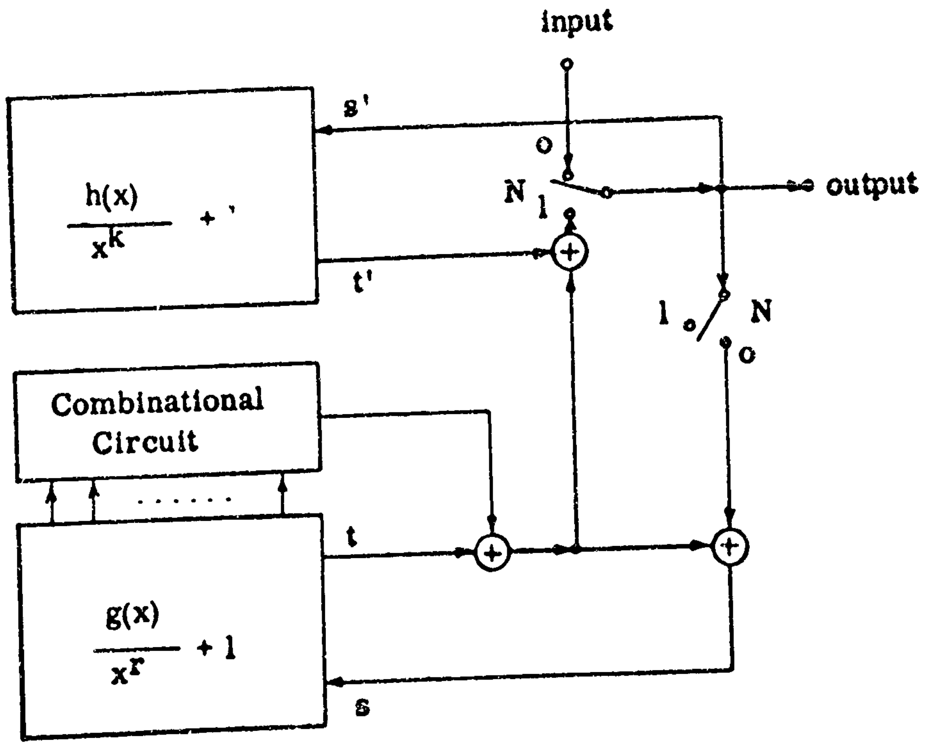


Figure A-10

CONCLUSION

We have shown that, instead of studying the shift-register contents, it is often convenient to study the input-output relationship of certain "building blocks". The equivalence relations obtained may help to present a clearer picture to those analyzing or designing the implementation of error-correcting codes.

An improved decoding circuit using a total of n shift register stages is also presented. The corrected code word is obtained after a delay of n bits, in comparison to 2n bits previously obtained in this type of decoding circuits.

REFERENCES

1. Abramson, N.M., "A Class of Systematic Codes for Non-Independent Error," IRE Transactions, IT-5, pp 150-157, December, 1959.

2. Elspas, B., "Design and Instrumentation of Error-Correcting Codes," Interim Technology Report, Stanford Research Institute, October, 1961.

3. Elspas, B., "The Theory of Autonomous Linear Sequential Networks," IRE Transactions, CT-6, pp 45-60, March 1959.

4. Fire, P., "A Class of Multiple-Error-Correcting Binary Codes for Non-Independent Errors," Report RSL-E-2, Sylvania Electric Products Incorporated, March 1959.

5. Huffman, D.A., "The Synthesis of Linear Sequential Coding Networks," Information Theory, Colin Cherry (Ed), Academic Press, New York, 1956.

6. Meggitt, J.E., "Error-Correcting Codes and Their Implementation for Data Transmission Systems," IRE Transactions, IT-7, pp 234-244, October, 1961.

7. Peterson, W.W., "Error Correcting Codes", MIT Technology Press, Cambridge, Mass. and Wiley, New York, New York; 1961.

8. Rudolph, L.D., and Mitchell, M.E., "Implementation of Decoders for Cyclic Codes," to be published in IEEE Transactions of Information Theory.

APPENDIX B

DUAL CODES VARIABLE REDUNDANCY CODES

INTRODUCTION

The theory of error-correcting codes has made a great deal of progress in recent years. Many interesting codes have been shown by means of their algebraic structures to possess various error-correcting (or detecting) capabilities suitable for different kinds of channels. An important class of codes is known as "cyclic codes" which enables one to use simple shift-register circuits with proper feed-back loops for encoding and decoding purposes.

There are situations when codes with different error-protecting capabilities are desired for the communication between a pair of fixed terminals. This may be either due to the existence of different classes of messages which demand different protections, or due to the change of real channels through which messages are transmitted.

While separate encoding and decoding circuits based independently on different codes are always possible, some sharing of facilities at both sending and receiving ends is much more preferable. Such code combinations are referred to as "variable-redundancy codes."

The class of dual codes falls naturally into the category of variable-redundancy codes. With $g(x)h(x) = x^n - 1$ where $g(x)$ and $h(x)$ are generating polynomials of the dual codes and n is the code length, $g(x)$ and $h(x)$ may generate codes with rather different redundancies. Actually, $g(x)$ and $h(x)$ can be chosen such that suitable different degrees of protection are obtained for different modes of operation. The implementation of dual codes is particularly interesting because of the possibility of using much of the same feed-back shift-register circuit in both encoding and decoding.

Because of its brevity, only binary codes are discussed throughout this report and polynomials with coefficients in the binary field are used for the code representation. Generalizations to cases with non-binary finite fields are possible.

For a cyclic (n, k) code C_g with generator polynomial

$$g(x) = x^r + a_{r-1}x^{r-1} + \ldots + a_1x + 1, \quad r = n-k, \tag{1}$$

we have $g(x) \mid x^n + 1$. The cyclic (n, r) code C_h generated by

$$h(x) = \frac{x^n + 1}{g(x)} = x^k + b_{k-1}x^{k-1} + \ldots + b_1x + 1 \tag{2}$$

is the dual code of C_g.

If g (x) is to possess any error-correcting ability at all, we must have n as the period of g(x), i.e., the smallest integer such that $g(x) \mid x^n + 1$. The same is true for h (x). The existence of $n' < n$ such that $g(x) \mid x^{n'} + 1$ implies that $x^{n'} + 1$ is a code word, and, therefore, the minimum distance is only 2. Such code cannot detect any set of all multiple independent errors up to a number > 1. However, the code can still be used to detect all bursts of r errors or less. The combination of correction and detection in different modes of operation with dual codes can sometimes be very practical. The existence of a feed-back channel, for instance, makes the error detection plus retransmission rather attractive in the normal mode of operation. Powerful correcting code, on the other hand, may be required when the other mode corresponds to some kind of emergency, and thus no delay is permissible.

Since the relation $g(x) \cdot h(x) = x^n + 1$ is a very strong restriction, it is not always possible to find g(x) such that both C_g and C_h are minimum-redundancy codes of their kinds for any given n and k. As a matter of fact, optimum C_g alone is sometimes not available. However, it is possible to choose g(x) for a considerable range of n and k, such that efficient codes are obtained in both modes. This becomes increasingly difficult as n increases. As a result, cases with large n become less attractive.

Before we illustrate how g(x) and h(x) can be chosen to obtain C_g and C_h with different combinations of error-protecting capabilities, we need the following basic theorems:

Theorem 1: Let g(x) be the generator of the cyclic code C_g of length n, thus $g(x) \mid x^n + 1$. Then $g^*(x)$, the reciprocal polynomial of g(x), generates an equivalent cyclic code C_g^* with the same maximum burst in the burst-error-protecting case and the same minimum distance in the independent-error-protecting case:

Proof: Since $g(x) \mid x^n + 1$, the roots of g(x) must be members of the multiplicative cyclic group containing all n'th roots of unity.

Since the inverse of each root of g(x) must be in the cyclic group of n elements, $g^*(x) \mid x^n + 1$. Furthermore, if n is the period of g(x), it is also the period of $g^*(x)$. For if the period of $g^*(n)$ is $n' \neq n$, then $n' \mid n$, and hence, $n' < n$. But then roots of g(x), being inverses of roots of $g^*(x)$, must be in the cyclic group of n' elements. This means $g(x) \mid x^{n'} + 1$, contradicting the assumption that $n > n'$ is the period of g(x).

The generator matrix of C_g can be written as

$$G = \begin{bmatrix} g(x) \\ x\,g(x) \\ x^2\,g(x) \\ ----- \\ x^{k-1}\,g(x) \end{bmatrix} \tag{5}$$

while the generator matrix of C_g^* can be written as

$$G^* = \begin{bmatrix} x^{k-1}\,g^*(x) \\ x^{k-2}\,g^*(x) \\ --- \\ x\,g^*(x) \\ g^*(x) \end{bmatrix} \tag{6}$$

It can be immediately seen that G^* is the same as G with the order of columns completely reversed. Since the same set of columns, as well as their adjacency relationships are maintained in both G and G^*, C_g and C_g^* must have the same independent-error and burst-error-correcting capability. The only requirement is that whenever $E(x)$ is an error pattern to be corrected or detected, so is $E^*(x)$.

Theorem 2[†]: Let α be any element of GF (2^m) of order e. Then the code consisting of all vectors $\{f(x)\}$ over the binary field for which

$$\alpha^{m_o}, \alpha^{m_o+s}, \alpha^{m_o+2s}, \ldots, \alpha^{m_o+(d-2)s}$$

are roots of $f(x)$ is a Bose-Chaudhuri code with minimum distance at least d, provided $(s, e) = 1$.

Proof: Let n be the code length and $d > 2$. Then, $\left(\alpha^{m_o}\right)^n = \left(\alpha^{m_o+s}\right)^n = 1$, and $\alpha^{sn} = 1$. It follows that $e \mid sn$, and therefore $e \mid n$, since $(s, e) = 1$. But $\alpha^e = 1$, and $\left(\alpha^{m_o+js}\right)^e = 1$, which means that the order of each of the above d-1 roots divides e, or $n \mid e$. Since $e \mid n$, and $n \mid e$, we have $n = e$.

The parity check matrix of the code can be written as

$$H = \begin{bmatrix} 1 & \alpha^{m_o} & \left(\alpha^{m_o}\right)^2 & --- & \left(\alpha^{m_o}\right)^{n-1} \\ 1 & \alpha^{m_o+s} & \left(\alpha^{m_o+s}\right)^2 & --- & \left(\alpha^{m_o+s}\right)^{n-1} \\ 1 & \alpha^{m_o+2s} & \left(\alpha^{m_o+2s}\right)^2 & --- & \left(\alpha^{m_o+2s}\right)^{n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \alpha^{m_o+(d-2)s} & \left(\alpha^{m_o+(d-2)s}\right)^2 & --- & \left(\alpha^{m_o+(d-2)s}\right)^{n-1} \end{bmatrix} \qquad (7)$$

[†] This generalization was pointed out by D. K. Ray-Chaudhuri of IBM Research Center.

The determinant of any set of d-1 columns of H is

$$\Delta = \alpha^{m_o(j_1 + j_2 + \dots + j_{d-1})} \begin{bmatrix} 1 & 1 & --- & 1 \\ \alpha^{sj_1} & \alpha^{sj_2} & --- & \alpha^{sj_{d-1}} \\ \left(\alpha^{sj_1}\right)^2 & \left(\alpha^{sj_2}\right)^2 & --- & \left(\alpha^{sj_{d-1}}\right)^2 \\ \vdots & \vdots & & \vdots \\ \left(\alpha^{sj_1}\right)^{d-2} & \left(\alpha^{sj_2}\right)^{d-2} & --- & \left(\alpha^{sj_{d-1}}\right)^{d-2} \end{bmatrix} \tag{8}$$

Consider any two columns corresponding to j_x and j_y We see that $\alpha^{sj_x} = \alpha^{sj_y}$ implies $\alpha^{s(j_y - j_x)} = 1$, or $e \mid s(j_y - j_x)$, which is impossible since $(s, e) = 1$, and $j_y - j_x < n = e$.

Since columns of (8) are all distinct, the van der Monde determinant never vanishes and the minimum distance of the code is no less than d.

CHOICE OF C_g and C_h

Interesting dual codes exist with various combinations of burst-error-correcting and independent-error-correcting capabilities.

A. BURST-ERROR-CORRECTING DUAL CODES:

Efficient burst-error-correcting dual codes are often possible by properly splitting the factors of $x^n + 1$. This can often be done by matching the right pairs of generator polynomials in a table of available efficient burst-error-correcting codes. For example, consider n = 21; we have:

$$x^{21} + 1 = (x^6 + x^4 + x^2 + x + 1)\ (x^6 + x^5 + x^4 + x^2 + 1)\ (x^3 + x^2 + 1)$$
$$(x^3 + x + 1)\ (x^2 + x + 1)\ (x + 1). \tag{9}$$

The code C_g generated by

$$g\ (x) = (x^6 + x^4 + x^2 + x + 1)\ (x + 1) \tag{10}$$

is an efficient triple-burst-error-correcting code as listed in Table III of Reference (6). The generator of the dual code

$$h\ (x) = \frac{x^{21} + 1}{g\ (x)} = (x^6 + x^5 + x^4 + x^2 + 1)\ (x^3 + x^2 + 1)\ (x^3 + x + 1)\ (x^2 + x + 1) \tag{11}$$

is not listed in the same table. But we readily see that $h^*(x)$, the reciprocal of h(x), is listed. By theorem 1, the code C_h generated by h(x) is also an efficient code. Some other burst-error-correcting dual codes found this way are show in the following table:

TABLE I

n	r	b	g (x)	k	b	h(x)
7	3	1	(13)	4	2	(15) (3)
15	6	3	(23) (7)	9	4	(31) (37) (3)
	7	3	(23) (7) (3)	8	4	(31) (37)
21	5	1	(13) (7)	16	8	(127) (165) (15) (3)
	6	2	(127)	15	7	(165) (13) (15) (7) (3)
	6	2	(13) (7) (3)	15	7	(127) (165) (15)
	7	3	(127) (3)	14	7	(165) (13) (15) (7)
	8	3	(127) (7)	13	6	(165) (13) (15) (3)
	8	4	(13) (15) (7)	13	5	(127) (165) (3)
	9	4	(13) (15) (7) (3)	12	5	(127) (165)
	9	4	(127) (7) (3)	12	6	(165) (13) (15)
	10	4	(127) (13) (3)	11	5	(165) (15) (7)

* Polynomials are given in octal representation. For example,

$$127 \rightarrow 1,010,111 \rightarrow x^6 + x^4 + x^2 + x + 1.$$

The reciprocal polynomials $g^*(x)$ and $h^*(x)$ also generate dual codes of the same redundancies as those generated by $g(x)$ and $h(x)$. They are not included in the above table.

B. INDEPENDENT-ERROR-CORRECTING DUAL CODES:

An interesting class of dual codes of Bose-Chaudhuri type can easily be found. A special case is the following:

Let α be a primitive element of GF (2^m) of order $e = 2^m - 1$, and let $g(x) = m_1(x)$, the minimal polynomial containing α. Then for all $m > 2$, both C_g and C_h has code length $n = 2^m - 1$. C_g has a minimum distance of 3, and C_h has a minimum distance at least 2^{m-1}. This can be seen from the fact that $h(x) = \frac{x^n + 1}{g(x)}$ contains $\alpha^3, \alpha^5, \alpha^7, \ldots, \alpha^{2^m - 1} = 1$, among its roots.

For cases in which C_g corrects more than single errors, more investigation is necessary to determine the error-correcting capability of C_h. To illustrate this, take α as a primitive element of GF (2^5). Here we list elements of GF (2^5) as powers of α such that powers of the roots of the same minimal polynomial are listed in the same row:

TABLE II

0					$m_0(x)$
1	2	4	8	16	$m_1(x)$
3	6	12	24	17	$m_3(x)$
5	10	20	9	18	$m_5(x)$
7	14	28	25	19	$m_7(x)$
11	22	13	26	21	$m_{11}(x)$
15	30	29	27	23	$m_{15}(x)$

Let $g(x) = m_1(x) \cdot m_3(x)$. Since it has α, α^2, α^3, and α^4 among its roots, the minimum distance d of C_g is at least 5.

Consider $h(x) = \frac{x^n + 1}{g(x)} = m_0(x) \cdot m_5(x) \cdot m_7(x) \cdot m_{11}(x) \cdot m_{15}(x)$. If we start with α^5 with $s' = 2$, we see that α^5, α^7, α^9, α^{11}, α^{13}, α^{15} are roots of h (x), indicating a minimum distance of 7. However, applying Theorem 2, a much better combination of m'_0 and s' is found as $m'_0 = 21$ and $s' = 5$. Thus h (x) is seen to have α^{21}, α^{26}, $\alpha^{31} = \alpha^0$, α^5, α^{10}, α^{15}, α^{20}, α^{25}, α^{30} among its roots, indicating a minimum distance d' of at least 10. Since α can be chosen as any element of proper order, g(x) is usually not unique. In this case, for instance, $m_1(x)$ may be chosen as $x^5 + x^2 + 1$, in which case we have

$$g(x) = (x^5 + x^2 + 1)(x^5 + x^4 + x^3 + x^2 + 1), \tag{12}$$

and

$$h(x) = (x^5 + x^4 + x^2 + x + 1)(x^5 + x^3 + x^2 + x + 1)(x^5 + x^4 + x^3 + x + 1)(x^5 + x^3 + 1)(x + 1) \tag{13}$$

Independent-error-correcting (or detecting) dual codes of various lengths and minimum distances can be found in a similar manner. Some of them are listed in the following table.

TABLE III

n	r	d	m	s	g(x)	k	d'	m'_0	s	h (x)
7	3	3	1	1	m(1)	4	4	3	2	m(0, 3)
15	4	3	1	1	m(1)	11	8	3	2	m(0, 3, 5, 7)
	5	4	0	1	m(0, 1)	10	7	3	2	m(3, 5, 7)
31	5	3	1	1	m(1)	26	16	3	2	m(0, 3, 5, 7, 11, 15)
	6	4	0	1	m(0, 1)	25	15	3	2	m(3, 5, 7, 11, 15)
	10	5	1	1	m(1, 3)	21	10	21	5	m(0, 5, 7, 11, 15)
	11	6	0	1	m(0, 1, 3)	20	7	5	2	m(5, 7, 11, 15)

TABLE III (Continued)

n	r	d	m	s	g(x)	k	d'	m_0'	s	h (x)
63	6	3	1	1	m(1)	57	32	3	2	m(0,3,5,7,9,11,13,15,21,23,27,31)
	7	4	0	1	m(0, 1)	56	31	3	2	m(3, 5, 7, 9, 11, 13, 15, 21, 23, 27, 31)
	12	5	1	1	m(1, 3)	51	16	53	5	m(0, 5, 7, 9, 11, 13, 15, 21, 23, 27, 31)
	13	6	0	1	m(0, 1, 3)	50	15	5	2	m(5, 7, 9, 11, 13, 15, 21, 23, 27, 31)
	18	7	1	1	m(1, 3, 5)	45	16	35	2	m(0, 7, 9, 11, 13, 15, 21, 23, 27, 31)
	19	8	0	1	m(0, 1,3,5)	44	15	35	2	m(7, 9, 11, 13, 15, 21, 23, 27, 31)

* $m(i_1, i_2, \ldots)$ is the minimum polynomial having $\alpha^{i_1}, \alpha^{i_2}, \ldots$ as roots. Thus, $m(i_1, i_2 \ldots) = m_{i_1}(x) \cdot m_{i_2}(x) \cdots$, Note that $m_1(x)$ is primitive in this table.

C. BURST-ERROR AND INDEPENDENT-ERROR-CORRECTING DUAL CODES

It is possible to combine the techniques described in previous sections to obtain a burst-error-correcting code C_g and an independent-error-correcting dual code C_h. A special case is the class of double-adjacent-error-correcting codes C_g of length 2^m-1. With $g(x) = (1 + x) f(x)$ where f(x) is primitive, we can identify it as $g(x) = m_0(x) \cdot m_1(x)$. The dual code generator h(x) has $\alpha^3, \alpha^5 \ldots \alpha^{2^m-3}$ among its roots, and thus C_h is a Bose-Chaudhuri code with minimum distance at least $2^{m-1}-1$.

Some other codes of this type found in the similar fashion are listed in the following table.

TABLE IV

n	r	b	g(x)	k	d'	m'_o	s'	$m_1(x)$	h(x)
15	5	2	(23) (3)	10	7	3	2	(23)	(37) (31) (7)
	6	3	(23) (7)	9	6	0	11		(37) (31) (3)
	8	4	(23) (37)	7	4	0	7		(31) (7) (3)
21	5	1	(13) (7)	16	8	0	1	(127)	(127) (165) (15) (3)
	6	2	(13) (7) (3)	15	7	1	1		(127) (165) (15)
	7	3	(127) (3)	14	5	3	2		(165) (15) (13) (7)
	8	3	(127) (7)	13	6	0	5		(165) (15) (13) (3)
31	6	2	(45) (3)	25	15	3	2	(45)	(75) (67) (57) (73) (51)
	10	4	(45) (75)	21	10	21	5		(67) (57) (73) (51) (3)
51	8	3	(433)	43	20	0	11	(763)	(763) (727) (471) (661) (637) (7) (3)
	10	4	(433) (7)	40	18	0	11		(763) (727) (471) (661) (637) (3)
63	7	2	(103) (3)	56	31	3	2	(103)	(127) (147) (111) (155) (133) (165) (163) (141) (15) (13) (7)
	9	3	(147) (13)	54	22	38	11		(103) (127) (111) (155) (133) (165) (163) (141) (15) (7) (3)
	12	5	(103) (13) (15)	51	18	23	5		(127) (147) (111) (155) (133) (165) (163) (141) (7) (3)
	14	6	(103) (111) (7)	49	14	33	5		(127) (147) (155) (133) (165) (163) (141) (15) (13) (3)
	15	7	(103) (133) (13)	48	14	43	5		(127) (147) (111) (155) (165) (163) (141) (15) (7) (3)
	18	8	(103) (127) (111)	45	13	9	2		(147) (155) (133) (165) (163) (141) (13) (15) (7) (3)
	19	9	(103) (147) (141) (3)	44	11	42	8		(127) (111) (155) (133) (165) (163) (13) (15) (7)

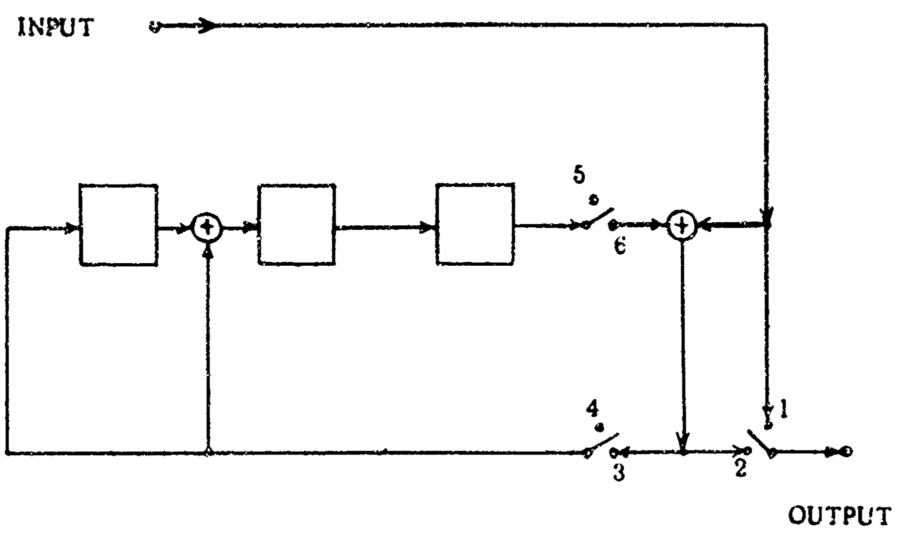
Let C_g be a separable cyclic (n, k) code generated by g(x) of degree r = n - k. Then a code word of C_g can be written as:

$$A(x) = x^r a(x) + r(x) \tag{14}$$

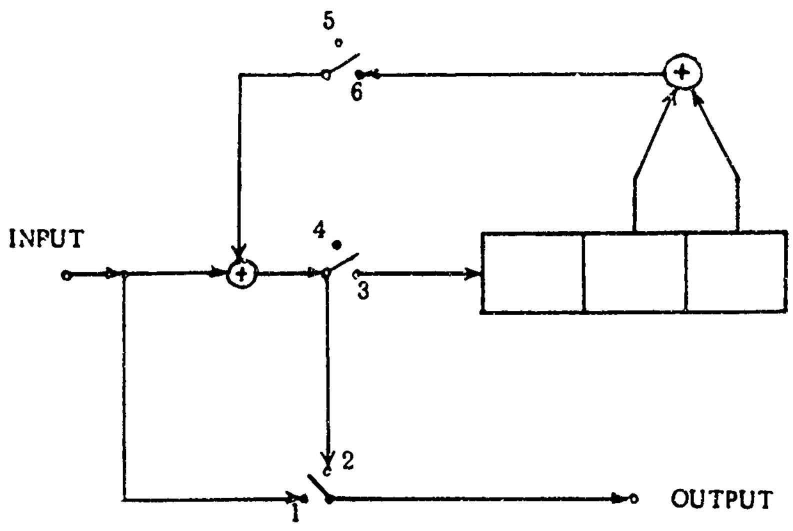
where a(x) represents the k bit message vector and r(x) is the residue of x^r a(x) modulo g(x). Thus $A(x) \equiv o \bmod g(x)$.

In order to obtain the residue r(x) with respect to a message a(x), it is possible to use (1) an r-stage shift-register with feed-back loops corresponding to g(x), or (2) a k-stage shift-register with feed-back loops corresponding to $h(x) = \frac{x^n + 1}{g(x)}$. With proper gating, the r-stage shift-register with feed-back loops corresponding to g(x) or h(x) alone can be used in the process of encoding and decoding for both of the dual codes C_g and C_h. We shall assume that $k > r = n-k$, and that an r-stage shift-register with feed-back loops corresponding to g(x) is used in the encoding and decoding circuitry. Code words are sent and received in the high-order-first manner.

There are two types of shift-register feed-back circuits, and both can be used for encoding or decoding purposes. The first type has feed-back loops connected to the mod-2 adders between adjacent stages of the shift-register. The feed-back loops correspond to the coefficients of g(x) with the high order ones at the output end of the shift-register. (Figure B-1A). The second type does not have mod-2 adders between stages, so the shift-register can often be replaced by a tapped delay-line. The feed-back loops also correspond to coefficients of g(x) with the high order ones at the input end of the shift-register (Figure B-1B).



(A)



(B)

Figure B-1

Figure B-1 shows the above two types of encoding circuits capable of operating in two modes:

A. MODE 1:

Encoding of the (n, k) code C_g generated by g(x) of degree $r = n-k$. In this special case $g(x) = x^3 + x + 1$, with $n = 7$, $k = 4$, $r = 3$. The time control at three switches is the same for both types:

Contact 1 and 3:	1st through kth bit-time until the last information bit is in the register.
Contact 2 and 4:	(k + 1)th through nth bit-time.
Contact 6:	1st through nth bit-time.

B. MODE 2:

Encoding of the (n, r) code C_h generated by $g'(x) = h(x) = \frac{x^n + 1}{g(x)}$ of degree k. In this special case $g'(x) = h(x) = x^4 + x^2 + x + 1$, or $h'(x) = g(x) = x^3 + x + 1$, with $n = 7$, $k = 4$, $r = 3$. The time control at three switches is the same for both types:

Contact 1 and 5:	1st through rth bit-time until the last information bit is in the register.
Contact 2 and 6:	(r + 1)th through nth bit-time.
Contact 3:	1st through nth bit-time.

Since both codes are separable, the information bits always go first directly to the channel while the feed-back shift-register circuit is ready to generate the check bits when the last information bit is in.

Figure B-2 shows the two types of decoding circuits capable of operating in two modes.

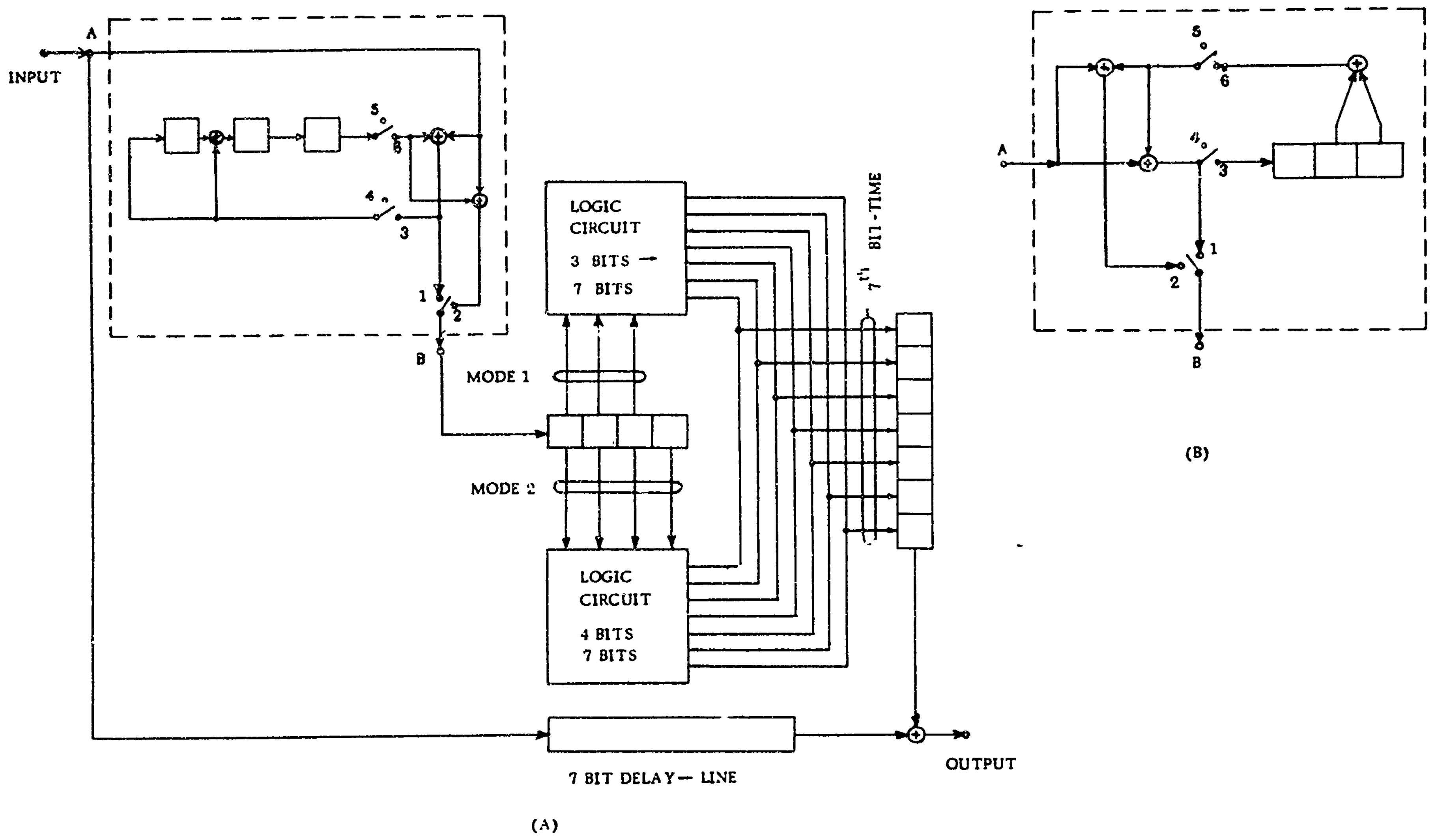

INPUT
A
5
6
4
3
1
2
B
MODE 1
MODE 2
LOGIC CIRCUIT 3 BITS → 7 BITS
LOGIC CIRCUIT 4 BITS 7 BITS
7th BIT-TIME
7 BIT DELAY — LINE
OUTPUT
(A)
(B)

Figure B-2

A. MODE 1:

Decoding of the (n, k) code C_g generated by $g(x) = x^3 + x + 1$. Time control at switches is the same for both types:

Contact 3:	1st through kth bit-time.
Contact 4:	(k + 1)th through nth bit-time.
Contact 1 and 6:	1st through nth bit-time.

B. MODE 2:

Decoding of the (n, r) code C_h generated by $g'(x) = h(x) = x^4 + x^2 + x + 1$. Time control at switches is again the same for both types:

Contact 5:	1st through rth bit-time.
Contact 6:	(r + 1)th through nth bit-time.
Contact 2 and 3:	1st through nth bit-time.

The basic feed-back shift-register circuit is the same as the encoding circuit. Check bits based on the received information bits are generated and compared with the received check bits. The difference can be identified as the error syndrome. To show this, suppose that $A(x) + E(x)$ is the received polynomial where $E(x)$ is the error polynomial. Let $E(x) = E_k(x) + E_r(x)$ with $E_k(x)$ containing the first k bits (high order terms of $E(x)$) and with $E_r(x)$ containing the last r bits of $E(x)$. Since r is the degree of $g(x)$, the syndrome of $E(x)$, which is the residue of $E(x)$ mod $g(x)$, is

$$r_e(x) = r_k(x) + E_r(x), \tag{15}$$

where

$$r_k(x) \equiv E_k(x) \qquad \text{mod } g(x) \tag{16}$$

It follows that

$$\begin{aligned}&[x^r a(x) + E_k(x)] + [r(x) + E_r(x)]\\ &\equiv r_k(x) + E_r(x) = r_e(x) \qquad \text{mod } g(x).\end{aligned} \tag{17}$$

The syndromes are mapped into error patterns by a logic circuit. The error pattern is then subtracted from the received polynomial to recover the original code word.

Since the mapping between syndromes and error patterns is arbitrary, the decoding circuits shown in Figure B-2 can be used for independent-error correction as well as burst-error correction.

CONCLUSIONS

The class of cyclic dual codes has been studied from the view-point of variable-redundancy codes. It is possible to obtain dual codes with various combinations of error protecting capabilities. The more interesting combinations are: (1) burst-error-correcting dual codes, (2) independent-error-correcting dual codes, and (3) burst-error-correcting and independent-error-correcting dual codes. Many efficient dual codes exist in each of the above three cases. Some of such codes with moderate lengths are tabulated.

The implementation of dual codes is based on the fact that it is possible to use the same feed-back shift-register circuit for both of the dual codes either when check bits are to be generated in encoding or when error syndromes are to be generated in decoding. Two types of general-purpose encoding and decoding circuits are shown to be applicable.

ACKNOWLEDGMENT

The author wishes to thank Drs. R. T. Chien and C. V. Freiman of the Thomas J. Watson Research Center, IBM Corporation, for their helpful discussions.

REFERENCES

1. Abramson, N. M., "A Class of Systematic Codes for Non-Independent Error," IRE Trans., IT-5, pp 150-157, December 1959.

2. Bose, R. C., and D. K. Ray-Chaudhuri, "On a Class of Error-Correcting Binary Group Codes," Information and Control, 3, pp 68-79, March 1960.

3. Bose, R. C., and D. K. Ray-Chaudhuri, "Further Results on Error Correcting Binary Group Codes," Information and Control, 3, pp 279-290, 1960.

4. Elspas, B., and Short, R. A. "A Note on Optimum Burst-Error Correcting Codes," IRE Trans., IT-8, pp 39-42, January 1962.

5. Elspas, B., "Design and Instrumentation of Error-Correcting Codes," Interim Tech. Report, Stanford Research Institute, October 1961.

6. Fire, P., "A Class of Multiple-Error-Correcting Binary Codes for Non-Independent Errors," Rept. RSL-E-2, Sylvania Electric Products Incorporated, March, 1959.

7. Hocquenghem, "Codes Correcteurs Derreurs," Chiffres 2, pp 147-156; September 1959.

8. Maggitt, J. E., "Error-Correcting Codes and their Implementation for Data Transmission Systems," IRE Trans. IT-7.

9. Peterson, W. W., "Error Correcting Codes," MIT Technology Press, Cambridge, Mass. and Wiley, New York, New York; 1961.

APPENDIX C

CYCLIC PRODUCT CODES WITH VARIABLE REDUNDANCY

I. INTRODUCTION

Practical situations in military and commercial communications systems often call for the transmission of messages with various degrees of urgency and reliability requirements. To meet such needs with economy and efficiency, one turns to coding techniques which employ a variable amount of redundancy that is adjusted to suit the situation at hand. Two outstanding problems that demand solutions in this area are, 1) the selection of suitable codes with variable redundancy, 2) the determination of efficient methods of informing the receiver of a mode change.

This paper is concerned with the first problem, that of selecting efficient, variable redundancy, and easily implementable codes. In particular, a class of codes, called the cyclic product codes, are investigated in detail. A cyclic product code is a cyclic code-pair that operates in a variable-redundancy mode. The high-redundancy code is generated with the generator polynomial $g(x) = g_1(x)\, g_2(x)$. The low-redundancy code is generated with the polynomial $g_1(x)$ only. We shall assume that the period of $g_1(x)\, g_2(x)$ is the same as that of $g_1(x)$. Several classes of cyclic product codes are constructed for the correction of independent errors, burst errors, and multiple bursts. They are all implementable with relatively simple hardware.

All discussions in the paper concern binary codes, the extension to non-binary cases is of a straight-forward nature in most cases; hence, it will not be treated in detail. For a general discussion on cyclic codes, see Peterson [1].

II. CYCLIC PRODUCT CODES

Let $g_1(x)$ be the generator polynomial of a cyclic code and n be the period of $g_1(x)$; then $g_1(x)$ divides $x^n + 1$. Furthermore, if $g_1(x)$ divides $x^k + 1$, k is a multiple of n. For most cases of interest n is odd, and therefore, $x^n + 1$

does not contain any repeated factors. As $g_1(x)\, g_2(x)$ has the same period n, one may conclude that $g_2(x)$ divides $x^n + 1$ and $g_1(x)$ is relatively prime to $g_2(x)$.

A. OPTIMUM CYCLIC PRODUCT CODE FOR CORRECTION OF BURST ERRORS

Suppose g(x) is a generator polynomial of a b-burst correcting code of length n, the period of g(x); then for any two error patterns $e_1(x)$ and $e_2(x)$ that are correctible

$$e_1(x) + x^i e_2(x) \neq 0 \quad (g(x)) \tag{1}$$

implies

$$e_1(x) + x^i e_2(x) \neq 0 \quad (x^n + 1) \tag{2}$$

where $\deg(e_1) \leq b - 1$, $\deg(e_2) \leq b - 1$.

The following theorem establishes a class of cyclic product codes for burst errors.

Theorem I. Let $g(x) = q_0(x)\, q_1(x)$ be the generator polynomial of a b - burst correcting code of length n, the period of g(x), and the degree of $q_0(x)$, Q, to be less than b. Then, $q_1(x)$ is the generator polynomial of a b_1 - burst correcting code of length n_1, the period of $q_1(x)$, where $b_1 \geq b - Q$ and n_1 divides n.

Proof: $q_1(x)$ generates a code of length n_1. Suppose this code does not have the ability to correct all b_1 - burst errors, then there exists two error patterns $e_1(x)$ and $e_2(x)$ such that

$$e_1(x) + x^i e_2(x) = 0 \quad (g_1(x)) \tag{3}$$

and

$$e_1(x) + x^i e_2(x) \not\equiv 0 \quad (x^{n_1} + 1) \tag{4}$$

where n_1 divides n, deg $(e_1) < l$, and deg $(e_2) < b_1$ if we multiply both sides of (3) by q_o,

$$q_o \left[e_1 (x) + x^i e_2 (x)\right] = 0 \qquad (g (x)). \tag{5}$$

As deg $(q_o e_1) < b_1 + Q = b$, and deg $(q_o e_2) < b_1 + Q = b$, by the error correcting properties of g (x) (5) implies

$$q_o \left[e_1 (x) + x^i e_2 (x)\right] = 0 \qquad (x^n + 1) \tag{6}$$

As both $q_o e_1$ and $q_o e_2$ are bursts of maximum length b and $2b < n$, it can easily be shown that (6) implies

$$q_o e_1 = q_o e_2$$

and n divides i. Hence, $e_1 = e_2$ and n_1 divide i. This is a contradiction to our assumption of the existence of e_1 and e_2 that satisfies both (3) and (4). The theorem therefore follows:

With the use of theorem I, one may take a b-burst correcting code and obtain from it a b_1 - burst correcting code by dropping a factor in the generating polynomial. The result is a pair of cyclic product codes that are suitable for variable redundancy applications.

Elspas and Short [2] had studied the construction of burst-error codes with minimum redundancy. For burst length b and r check bits their codes have a length of $n = 2^{r - b + 1} - 1$. Tables of minimum redundancy code have been presented for n up to 4095 and b up to four. These codes are very suitable for operating in the variable redundancy mode as cyclic product codes.

Example: the code generated by

$$g_4(x) = (1+x)\ (1+x+x^2)\ (x+x^2+x^3+x^4+x^6+x^{10}+x^{12})$$

corrects all bursts of four bits or less with n = 4095. By dropping some factors we may obtain a burst-3 correcting code with

$$g_3(x) = (1+x+x^2)\ (1+x^2+x^3+x^4+x^6+x^{10}+10^{12})$$

or a burst - 2 correcting code with

$$g_2(x) = (1+x)\,(1+x+x^3+x^4+x^6+x^{10}+x^{12})$$

or a burst - 1 correcting code with

$$g_1(x) = (1+x^2+x^3+x^4+x^6+x^{10}+x^{12}).$$

The fact that these codes do error-correction as claimed follows directly from Theorem I. We list in Table I a class of codes selected from Elspas and Short [2]. Those codes listed are suitable cyclic product codes. It should be noted that Theorem I could also be used to limit the amount of labor required in searching for minimum redundancy codes in general.

TABLE I

CYCLIC PRODUCT CODES FOR BURST ERRORS

n	r	g (x) (Listing Powers with Units Coefficient for Each Factor)
1023	13	(01) (012) (0235, 10)
		(01) (012) (012467, 10)
		(01) (012) (023458, 10)
		(01) (012) (012369, 10)
		(01) (012) (013469, 10)
4095	15	(01) (012) (0347, 12)
		(01) (012) (012459, 12)
		(01) (012) (034589, 12)
		(01) (012) (01234589, 12)
		(01) (012) (02346, 10, 12)
		(01) (012) (02348, 10, 12)
		(01) (012) (01478, 10, 12)
		(01) (012) (0234678, 10, 12)
		(01) (012) (01269, 10, 12)
		(01) (012) (01246, 11, 12)
		(01) (012) (0125789, 11, 12)
		(01) (012) (0124, 10, 11, 12)
		(01) (012) (012347, 10, 11, 12)

B.- OTHER CYCLIC PRODUCT CODES FOR BURST-ERROR-CORRECTION

The Fire Codes [3] are defined by the generator polynomial

$$(x^c + 1) \quad p(x)$$

where p(x) is a polynomial of degree $\geq b$ and period e, and $c \geq 2b - 1$. This code is capable of correcting a single burst of maximum length b with a block length n which is L.C.M. (c, e). For most cases in practice, c is taken to be $2b - 1$ and p(x) chosen to be a primitive irreducible polynomial of degree b; hence the block length is $c(2^b - 1)$.

It is relatively easy to see that the class of cyclic product codes may be constructed by taking the pair

$$g_1(x) \; g_2(x) = (x^c + 1) \; p(x)$$

$$g_1(x) = (x^{c_1} + 1) \; p(x)$$

where c_1 divides c. The code generated with $g_1(x)$ alone will correct a shorter burst. However, as it is longer than the normal length of a Fire code, it is closer to optimality in redundancy.

Another way to generate a class of cyclic product codes is by applying Theorem I. This will be illustrated by the following pair of generator polynomials:

$$g_1(x) = (x^7 + x^6 + x^4 + x^3 + x + 1) \; (x^5 + x^2 + 1)$$

$$g_1(x)g_2(x) = (x^9 + 1) \; (x^5 + x^2 + 1)$$

Thus, $g_1(x)$ generates a code for correcting burst-3 which has a length of $9 \times 31 = 279$. $g_1(x) \; g_2(x)$ generates a burst - 5 Fire code of length 279. It should be noted that $(x^7 + 1) \; (x^5 + x^2 + 1)$ generates a burst - 4 Fire code of length 217. Comparing with the result obtained in II A by applying Theorem I, it is clear that the most advantageous situation for applying Theorem I is when the higher redundancy code of the pair is close to optimum.

A third method to approach the cyclic product codes for burst correction is to select a code pair with the same generator polynomials which give different code lengths. This may be due to the difference in their natural lengths, or can be a result of shortening the code, or both.

$$g(x) = (1 + x)^2\ p(x)$$

where

$$p(x) = p_o + p_1 x + --- + p_{m-1}\ x^{m-1} + p_m\ x^m,\ m \text{ odd},$$

with $p_i = 0$ if $i = 1$

$p_i = 1$ otherwise,

generates a class of optimum shortened cyclic burst - 3 codes of length $2^m - m$ Ref. [4]. These same polynomials generate burst - 2 codes of lengths $2\ (2^m - 1)$. The polynomial $g_1\ (x) = (1 + x)\ p\ (x)$ generates burst - 2 codes of length $2^m - 1$. Results on shortened codes as variable redundancy codes will be reported separately.

C. CYCLIC PRODUCT CODES FOR CORRECTING INDEPENDENT ERRORS AND MULTIPLE BURSTS

Independently, Bose-Chaudhuri [5] and Hocquenghem [6] had suggested a very general class of cyclic codes for correcting independent errors. The BCH codes are most conveniently described in terms of the finite field GF (2^m). A t -error-correcting BCH code is generated by the polynomial g(x) such that

$$g(x^i) = 0 \qquad i = 1, 3, ---, 2t - 1$$

where α is a primitive root of GF (2^m). The length of such a code is $2^m - 1$. The BCH codes are most suitable as cyclic product codes as we may, in general, write

$$g(x) = \text{LCM}\ (m_1\ (x)\ m_2\ (x)\ ---\ m_{2t}\ (x))$$

where $m_i\ (x)$ is the minimal polynomial of α^i. As a special case of the general BCH code over GF (q^m), Reed and Solomon suggested a class of codes where the roots of the polynomial are not taken from the extension field, but from the ground field itself. The RS code is of length $2^m - 1$, where the generator polynomial is

$$g(x) = (x - \alpha)(x - \alpha^2) \cdots (x - \alpha^{2t}).$$

The RS code will have minimum distance $d = 2t + 1$ and will correct t independent errors. The most useful way a RS code can be practiced, however, is by using it as multiple-burst-correcting codes. For a t-error-correcting code a RS code can correct t in-phase bursts of maximum length m in a code length $(2^m - 1)$ m. For the same reason as in the case of BCH codes, the RS codes are cyclic product codes in the sense that the more factors the generator polynomial have, the more bursts the code can correct. It is also true that one may use the multiple bursts structure to correct a single long burst. And sometimes it is better than a Fire code of corresponding length.

As an illustration, a RS code with symbols from GF (2^7) and $t = 4$ could correct all bursts of length 22 or less. Its length is 889. To obtain a RS code of the same length but correcting a short burst, simply let $t = 3, 2, 1$, and we obtain code for correcting $b = 15$, $b = 8$, $b = 1$, respectively.

III. IMPLEMENTATION OF CYCLIC PRODUCT CODES

A. GENERAL CONSIDERATIONS

Let $g_1(x)$ be the generator polynomial of a (n, k_1) code C_1, and $g(x) = g_1(x)\, g_2(x)$ be the generator polynomial of a (n, k) code C where the degree of $g_1(x)$ is $r_1 = n - k_1$, and the degree of $g(x)$ is $r = n - k$. To illustrate the implementation of the product codes, a specific example will be used throughout this report for encoding and burst-error correcting. The cyclic product code-pair operate in the following two modes:

(1) Mode 1: C_1 is a (15, 11) code generated by

$$g_1(x) = x^4 + x + 1,$$

with

$$h_1(x) = \frac{x^{15} + 1}{g_1(x)} = x^{11} + x^8 + x^7 + x^5 + x^3 + x^2 + x + 1.$$

This code corrects single errors.

(2) Mode 2: C is a (15, 9) code generated by

$$\begin{aligned} g(x) &= g_1(x)\, g_2(x) \\ &= (x^4 + x + 1)\,(x^2 + x + 1) \end{aligned}$$

with

$$h(x) = \frac{x^{15} + 1}{g(x)} = x^9 + x^8 + x^5 + x^4 + x^3 + 1.$$

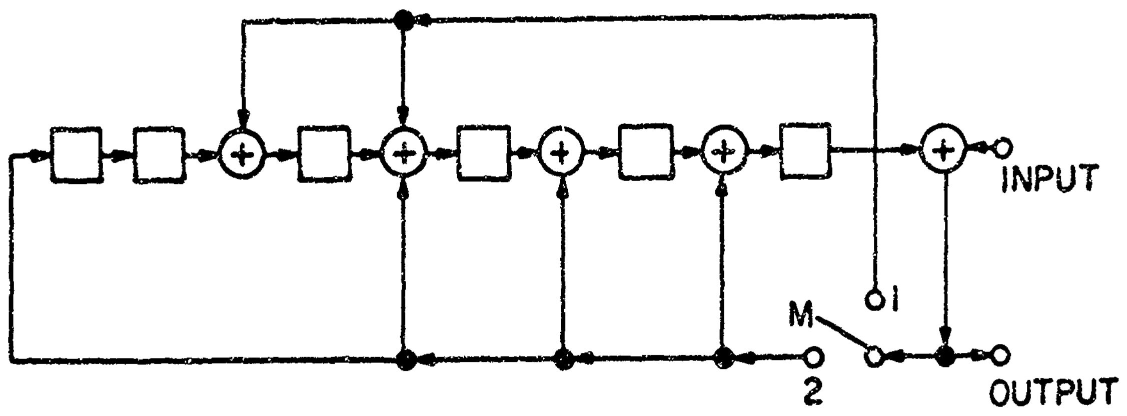
This code corrects burst errors up to length = 3.

The above product code-pair has the property that both C_1 and C are optimum codes. Longer product codes with the same optimality property can be obtained from Table I.

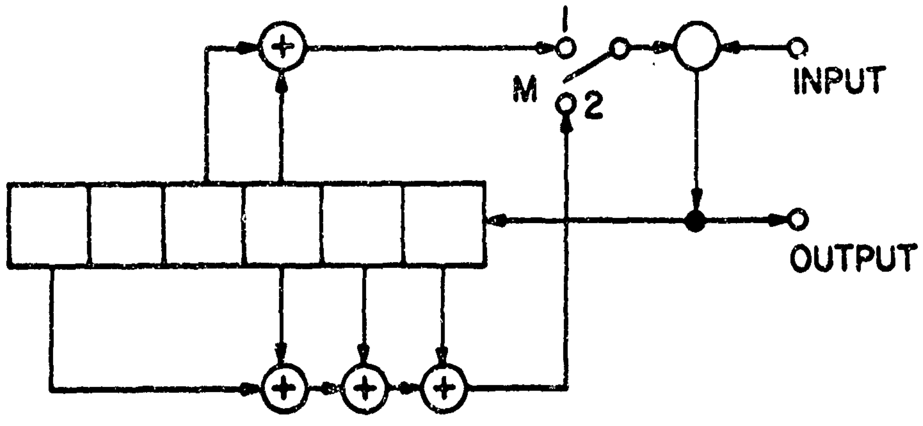
To implement a cyclic product code capable of operating in two or more modes, we should take advantage of the properties of the product code so that some sharing of the circuitry when operated in different modes is possible. An immediate approach is to share the shift-registers in different modes and switch the interconnection from one configuration to another to obtain a change in its function required by the different mode of operation.

Figure C-1 shows two division circuits designed according to the above philosophy. Switch M is at position "1" for Mode 1 and at position "2" for Mode 2. Such division circuits can be parts of the encoding or decoding circuitry.

It can be seen from the circuits of Figure C-1 that, because of an extra mode of operation, some new mod-2 adders must be installed, and some mod-2 adders originally with two inputs now become mod-2 adders with three inputs. Generally speaking, the complication thus introduced is roughly proportioned to the degree of $g_1(x)$ or g(x). To apply this direct sharing approach to the type of decoding circuits which uses a total of n shift-registers (second part of Section III-C) introduces further complications and makes it rather uninteresting.



(A)



(B)

Figure C-1

A better approach, which is both practically more appealing and theoretically more interesting, is to combine the basic building blocks corresponding to $g_1(x)$ and $g_2(x)$ in obtaining the building block corresponding to $g(x) = g_1(x)\,g_2(x)$. Such basic building blocks can be connected to make simple multiplication or division circuits, as well as various types of encoding and decoding circuits in general. With proper switching elements in the circuits, different modes of operation can be obtained. The complication introduced due to multiple modes of operation can be kept at its minimum and is independent of the sizes of $g_1(x)$ and $g_2(x)$.

A "basic block" corresponding to g(x) here refers to a shift-register circuit whose input and output sequence is related by the transfer function $\frac{t}{s} = \frac{g(x)}{x^r} + 1 = g^*(D) + 1$ where $x^{-1} = D$ is considered as a delay operator. See Ref. [7].

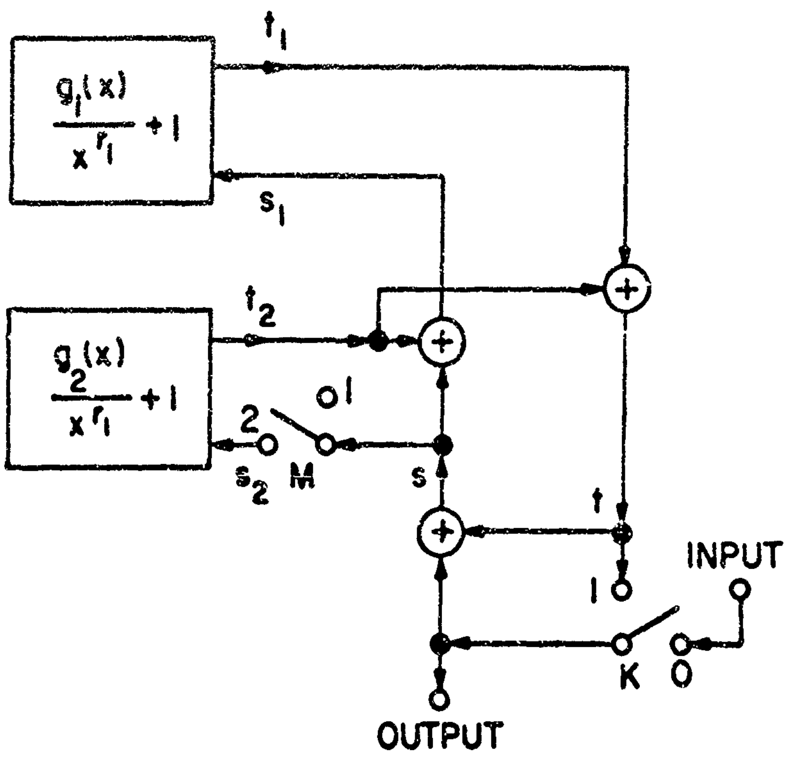


Figure C-2

Figure C-2 shows a general encoding circuit for any cyclic product code-pair. The switches are controlled according to the following:

Mode 1: Generator polynomial = $g_1(x)$

$M = 1$ for all n bits

$K = 0$ for the first k_1 bits

1 for the remaining r_1 bits

Mode 2: Generator polynomial = $g(x) = g_1(x)\ g_2(x)$

$M = 2$ for all n bits

$K = 0$ for the first k bits

1 for the remaining r bits.

The code words obtained are separable. That is, a k -bit message m(x) is coded into

$$C(x) = x^r\ m(x)\ +\ r(x)$$

where

$$r(x) \equiv x^r\ m(x) \qquad \mod g(x)$$

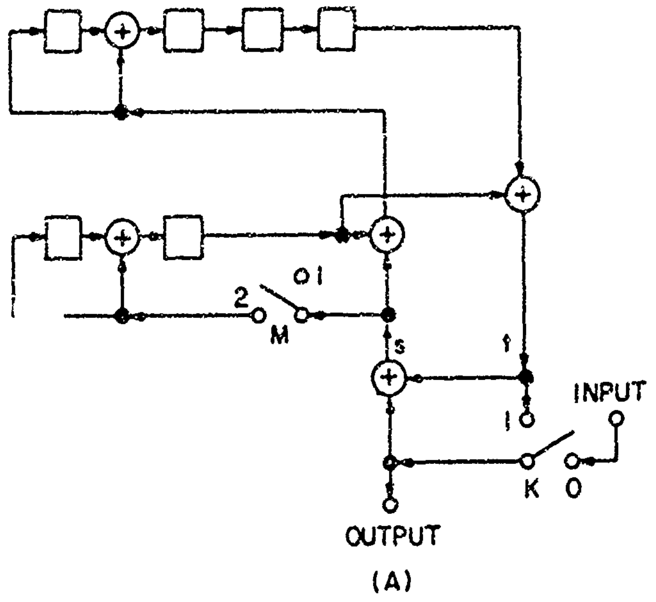
has degree no more than r - 1.

Note that the relationship between s and t in Figure C-2 is described by the following transfer function

$$\frac{t}{s} = \frac{g(x)}{x^r} + 1 = g^*\,(D) + 1$$

where $D = x^{-1}$ is the delay operator and g^* (D) is the reciprocal polynomial of g(D).

Figure C-3 shows two encoding circuits for the specific cyclic product code-pair described earlier.



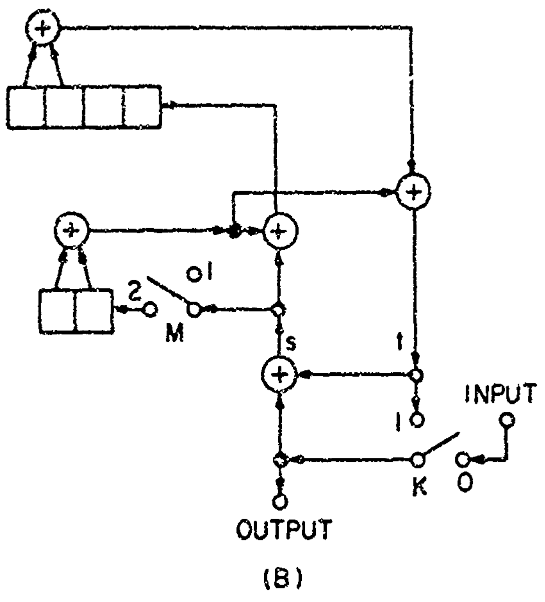


Figure C-3

C. DECODING CIRCUITS FOR BURST-ERROR-CORRECTING

Two types of decoding circuits for burst-error-correcting product codes will be described in this section:

(1) The first type of decoding circuit contains an n-stage buffer storage and a division circuit corresponding to g(x). The vector with shift-register contents as components forms an error syndrome, which does not necessarily correspond to the residue of the error polynomial modulo g(x). The basic approach is the same as that of Peterson [1] and Meggitt [8].

In order that the circuit is capable of correcting burst errors in different modes, the division circuit must contain individual building blocks corresponding to $g_1(x)$ and $g_2(x)$ so that one of them can be disconnected in the low-redundancy mode. The syndromes should go through an error cycle and should be recognized when the first bit of a correctible burst error is ready to leave the n-stage buffer storage.

Figure C-4 shows two decoding circuits for the same product code with $g(x) = (x^4 + x + 1)(x^2 + x + 1)$. Note that since the difference in the maximum correctible burst lengths is the same as the difference in the degrees of generator polynomials in two modes, the same zero-detecting circuit can be used in both modes. This applies to all burst-error-correcting product codes which

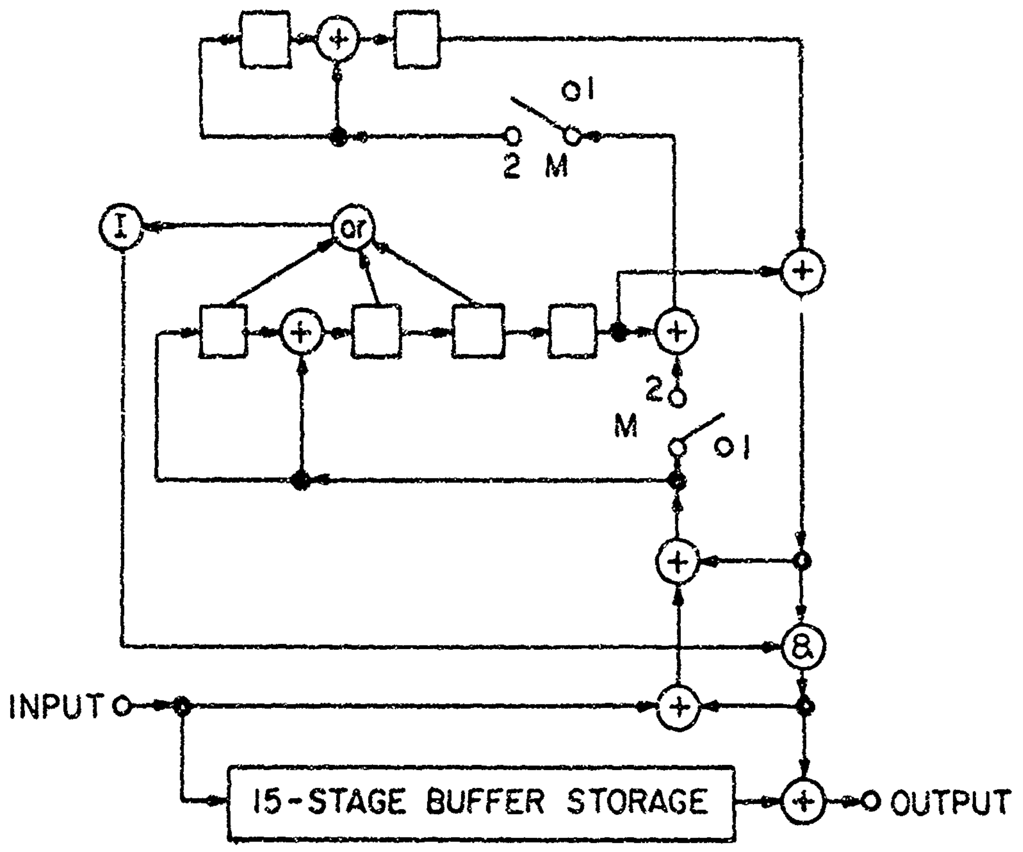


Figure C-4-A

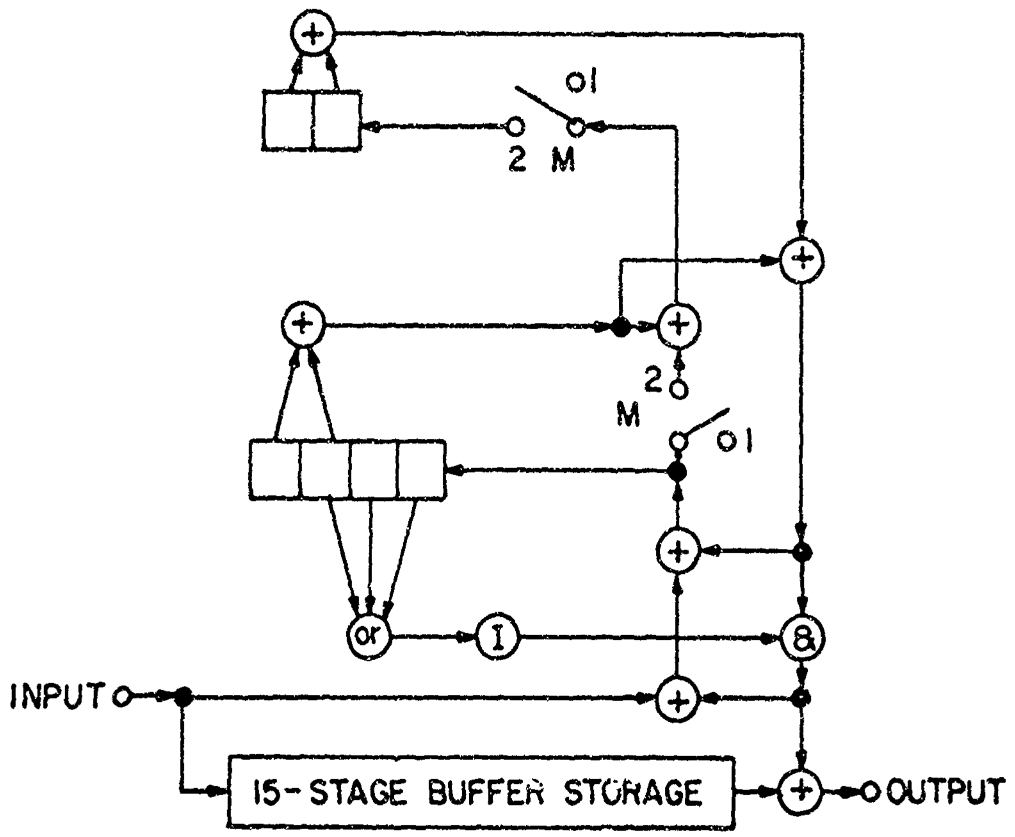


Figure C-4-B

operate according to Theorem 1, such as those obtained from Table I. In general, the zero detecting circuit may require a different set of input connections in different modes.

In terms of matrix notation, the shift-register contents are syndromes in the form of row vectors determined by multiplying the received vector (a code vector plus an error vector) a d the transposed parity-check matrix:

$$S = (C + E)\ H^T = EH^T$$

The parity-check matrix corresponding to the decoding circuit of Figure C-4-A is

$$H_a = \left[\begin{array}{ccccccccccccccc}
1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\
\hdashline
0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1
\end{array}\right]$$

and the parity-check matrix corresponding to the decoding circuit of Figure C-4-B is

$$H_b = \left[\begin{array}{ccccccccccccccc}
1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\
\hdashline
0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1
\end{array}\right]$$

(2) The second type of decoding circuits also contains a division circuit corresponding to g(x). However, a k-stage shift-register circuit corresponding to $h(x) = (x^n + 1)/g(x)$ takes the place of the n-stage buffer storage. The decoded (or the original, in case of detecting an uncorrectible error) message is obtained after a delay of n bits instead of 2n bits in the error-trapping approach of Mitchell [9].

To meet the need of two-mode operation. The decoding circuit must contain the building blocks corresponding to $g_1(x)$, $g_2(x)$, and $h(x) = (x^n + 1)/g_1(x)\,g_2(x)$. In the high-redundancy mode, the building blocks of $g_1(x)$ and $g_2(x)$ are combined to form the equivalent block corresponding to $g(x) = g_1(x)\,g_2(x)$. In the low-redundancy mode, the building blocks of h(x) and g(x) are combined to form the equivalent block corresponding to $h_1(x) = (x^n + 1)/g_1(x)$.

Figures C-5 and C-6 are two decoding circuits for the specific example of cyclic product code pair given earlier. All switches designated by M are at position "1" for mode 1 and at position "2" for mode 2. All switches designated by N are at position "0" for the first n bits and then at position "1" for the remaining n bits in both modes. The relationship between s_h and t_h in both circuits is described by the transfer function

$$\left(\frac{t_h}{s_h}\right)_1 = \frac{h_1(x)}{x^k} + 1 = \frac{h(x) \cdot g_2(x)}{x^{(k+r_2)}} + 1$$

in mode 1, and by

$$\left(\frac{t_h}{s_h}\right)_2 = \frac{h(x)}{x^k} + 1$$

in mode 2. The relationship between s_g and t_g in both circuits is described by the transfer function

$$\left(\frac{t_g}{s_g}\right)_1 = \frac{g_1(x)}{x^{r_1}} + 1$$

in mode 1, and by

$$\left(\frac{t_g}{s_g}\right)_1 = \frac{g(x)}{x^r} + 1 = \frac{g_1(x)\,g_2(x)}{x^{(r_1 + r_2)}} + 1$$

in mode 2. The corresponding building blocks in Figures C-5 and C-6 are equivalent and are hence interchangeable.

Figure C-5

C 20.

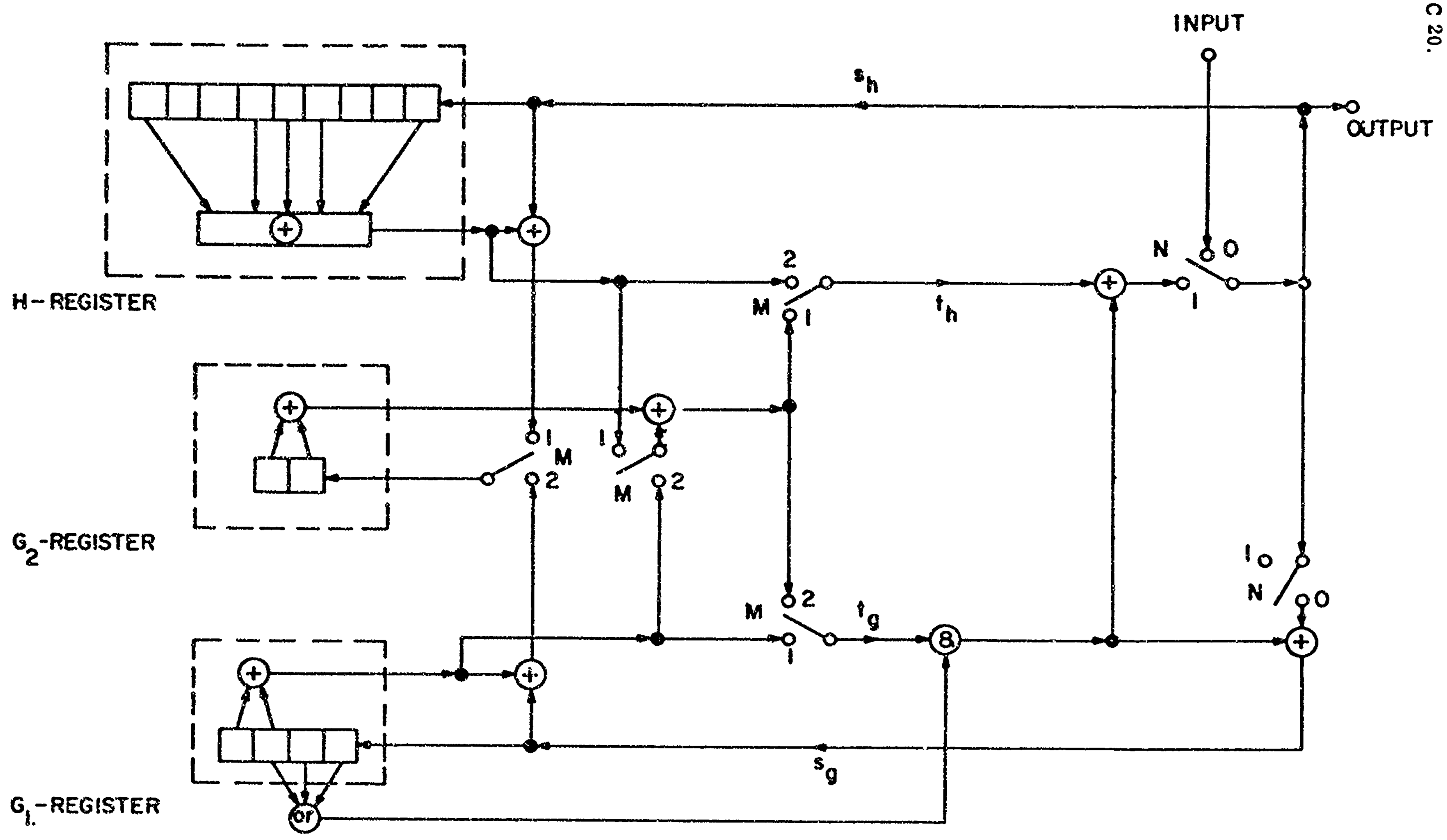


Figure C-5

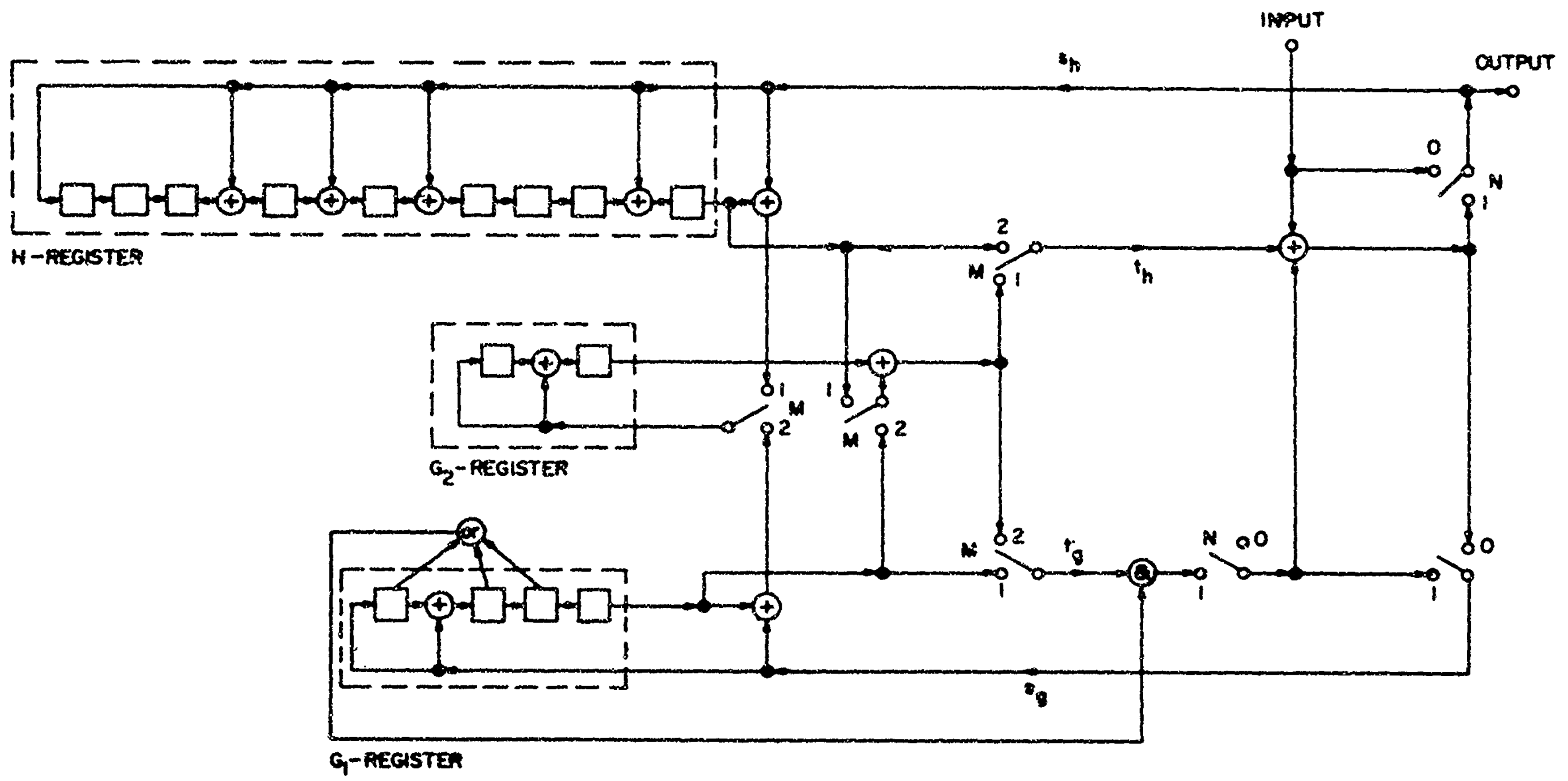

INPUT
OUTPUT
s_h
t_h
t_g
s_g
H-REGISTER
G_2-REGISTER
G_1-REGISTER
or
M
N
0
1
2

Figure C-6

These two decoding circuits operate on slightly different principles for the first n bits. In both cases, however, the error syndrome is obtained at the end of first n bits in the shift-register corresponding to the generator polynomial. For the second n-bit cycle, both circuits are connected in such a way that the received message can be regenerated. At the same time, the syndromes go through their error cycle and are recognized when the first bit of a correctible burst error is about to reappear at the output. By complementing the next bit in the output and subtracting the corresponding syndrome from the shift-register contents, the correctible errors are corrected sequentially. For more discussions, see Ref. [7].

D. DECODING CIRCUITS FOR INDEPENDENT-ERROR-CORRECTING

Each of the decoding circuits described previously (Figures C-4, C-5, and C-6) can be used for independent-error-correction if a combinational circuit is used for syndrome recognizer. However, this may be limited to cases where the minimum distance of the code is relatively small. For codes with larger distance the size of the combinational syndrome recognizer may become prohibitively large.

For Bose-Chaudhuri-Hocquenghem codes, a cyclic decoder can be used. This improved decoding procedure takes advantage of the cyclic property of the code, and the errors are corrected sequentially at the output.

Let g(x) be the generator polynomial of a t-error-correcting code of length $2^m - 1$ such that g(x) is the minimum polynomial containing, α, α^3, α^5, ---, α^{2t-1} as roots. Upon receiving a polynomial r(x), syndromes $S_1 = r(\alpha)$, $S_3 = r(\alpha^3)$, ---, $S_{2t-1} = r(\alpha^{2t-1})$ can be used to obtain the elementary symmetric functions σ_1, σ_2, ---, σ_t. The non-zero roots of the polynomial

$$\Sigma(x) = x^t + \sigma_1 x^{t-1} + \text{---} + \sigma_{t-1} x + \sigma_t$$

represent the errors.

Instead of the previous exhaustive approach in finding the roots, the following procedure is carried out: The polynomial is sequentially transformed such that at i'th step, we have

$$\Sigma^{(i)}(x) = x^t + \sigma_1 \alpha^i x^{t-1} + \text{---} + \sigma_{t-1} \alpha^{(t-1)i} x + \sigma_t \alpha^{ti}.$$

Therefore, if β is a root of $\Sigma(x)$, $\alpha^i\beta$ is a root of $\Sigma^{(i)}(x)$. Consider an error at the j'th position, thus $\beta = \alpha^{j-1}$. After $i = n - j + 1$ transformations, $\alpha^{i+j-1} = \alpha^o = 1$ becomes a root of $\Sigma^{n-j+1}(x)$. By checking whether "1" is a root of $\Sigma^{(i)}(x)$ at each step, the bit in error can be corrected sequentially at the output.

Figure C-7 shows a schematic block diagram of the cyclic decoding procedure. It can be seen that if the decoder is designed for the code with generator polynomial $g(x) = g_1(x)\, g_2(x)$, the low-redundancy mode of operation with generator polynomial $g_1(x)$ requires essentially no change of the circuit. For if $g_1(x)$ generates a t_1 - error-correcting code $\sigma_i = o$ for $i > t_1$.

Figure C-8 shows another cyclic decoder for BCH codes. Here, the transformed syndromes are fed directly to a combinational circuit which checks the equivalent condition for "1" to be a root of $\Sigma^{(i)}(x)$. Again, little change is necessary to make a decoding circuit corresponding to high-redundancy mode work in the low-redundancy mode. The part of the combinational circuit which checks the high order error greater than t_1 can be inactivated in the low-redundancy mode. Whether this type of decoding circuit is suitable, again, depends upon the size of the combinational circuit used.

For more discussions of cyclic decoding approach for BCH codes in general, see Ref. [10] and [11].

CONCLUSION

This paper presents original results on the theory and implementation of cyclic product codes. Various classes of codes are constructed for the correction of burst errors, independent errors, and multiple bursts. Decoding circuits with n - shift register stages, a theoretical minimum, are presented for burst errors. An algebraic-cyclic procedure is suggested for correction of independent errors and multiple bursts.

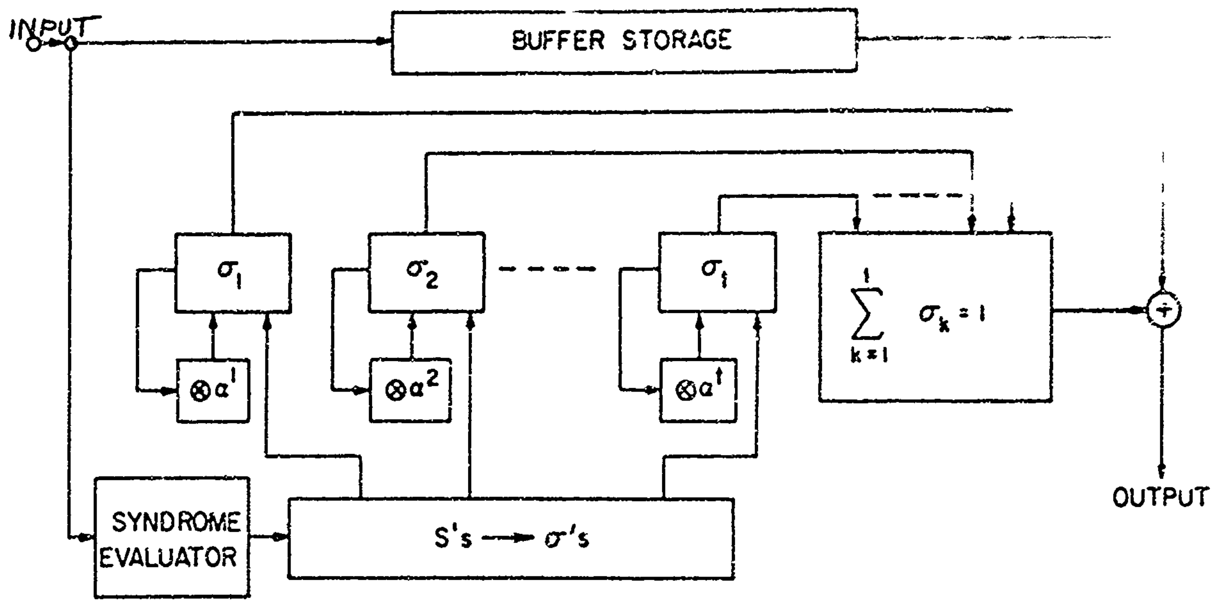


Figure C-7

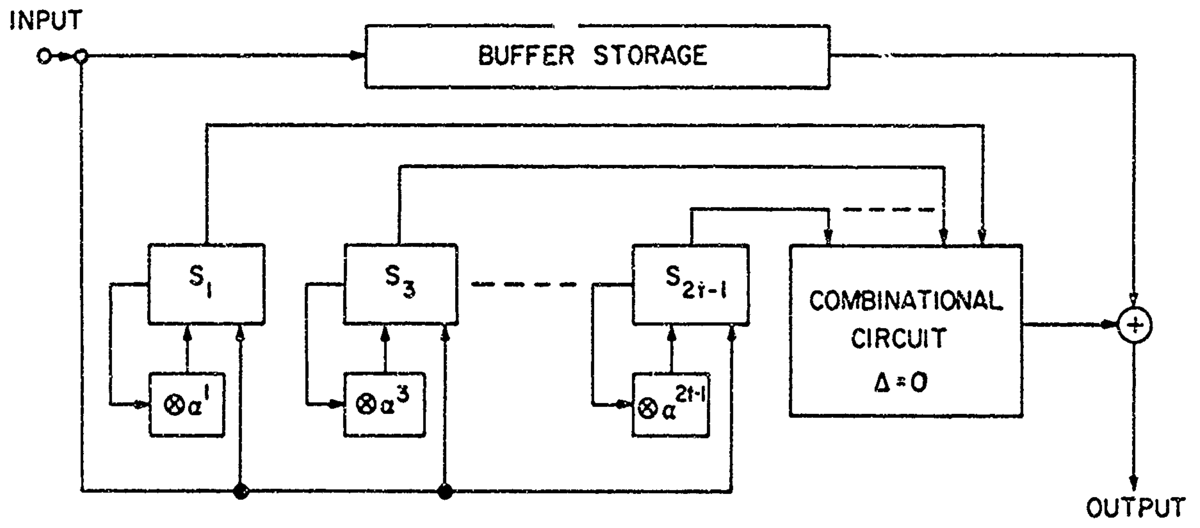


Figure C-8

REFERENCES

1. Peterson, W.W., Error Correcting Codes, MIT Technology Press, Cambridge, Mass., and Wiley, New York, New York; 1961.

2. Elspas, B., and Short, R.A., "A Note on Optimum Burst-Error Correcting Codes," IRE Trans., IT-8, pp. 39-42, January 1962.

3. Fire, P., "A Class of Multiple-Error-Correcting Binary Codes for Non-Independent Errors," Rept. RSL-E-2, Sylvania Electric Products Inc., March, 1959.

4. Chien, R.T., "Shortened Cyclic Codes for Burst Correction," IBM Research Note NC-349, 1964.

5. Bose, R.C., and D.K. Ray-Chaudhuri, "On a Class of Error-Correcting Binary Group Codes," Information and Control, 3, pp. 68-79, March, 1960.

6. Hocquenghem, "Codes Correcteurs Derreurs," Chiffres 2, pp. 147-156; September 1959.

7. Tang, D.T., "On the Implementation of Error-Correcting Codes," IBM Research Report, 1964.

8. Meggitt, J.E., "Error-Correcting Codes and Their Implementation for Data Transmission Systems," IRE Trans., IT-7, pp. 234-244, October, 1961.

9. Rudolph, L.D., and Mitchell, M.E., "Implementation of Decoders for Cyclic Codes," to be published in IEEE Trans. Information Theory.

10. Chien, R.T., "A Cyclic Decoder for Bose-Chaudhuri-Hocquenghem Codes," IBM Research Report RC-1063, 1963.

11. Chien, R.T., "On the Instrumentation of Bose-Chaudhuri-Hocquenghem Codes," Research Note NC-323, 1963.

APPENDIX D

ON VARIABLE LENGTH CODES AS VARIABLE REDUNDANCY CODES

In order that a communication system can have more than a fixed amount of error protection, different codes may be used in different modes of operation. Dual codes [7] and product codes [2] are two major classes studied which offer variable redundancy code-pairs with minimum additional implementation. In both cases, the length of a message block is the same in different modes; i.e., the periods of the generator polynomials used in different modes are the same.

By allowing the block length to vary from one mode to another, some interesting code combinations may be constructed. These will be investigated in the following sections.

1. PRODUCT CODES WITH DIFFERENT NATURAL LENGTHS

Various code combinations are possible. Some of them are demonstrated in the following examples:

Example 1: Let C_1 be a Fire code generated by

$$g_1(x) = (x^7 + 1)(x^5 + x^2 + 1). \qquad (1)$$

Then C_1 is a burst-4 correcting code of length $n_1 = 217$. This code can be lengthened by multiplying an irreducible factor of degree four or higher [4]. The code C generated by

$$g(x) = g_1(x)(x^4 + x + 1) \qquad (2)$$

is at least a burst-4 correcting code, but the length is $n = 217 \times 15 = 3255$. C is the low-redundancy code and C_1 is the high-redundancy code.

Example 2: Let C_1 be the code generated by

$$g_1(x) = x^6 + x^3 + 1, \tag{3}$$

Then C_1 is a burst-3 correcting code of length $n_1 = 9$. The code C generated by

$$g(x) = g_1(x)\ (x^6 + x + 1)\ (x^6 + x^4 + x^2 + x + 1)\ (x^6 + x^5 + x^2 + x + 1) \tag{4}$$

is a Bose-Chaudhuri-Hocquenghem code of length $n = 63$ which corrects at least four independent errors.

2. DUAL CODES WITH DIFFERENT NATURAL LENGTHS

We show the following example.

Example 3: Let C be the code generated by

$$g(x) = x^3 + x + 1.$$

Then C is a single-error-correcting code of length $n = 7$. The code C' generated by

$$h(x) = \frac{x^{63} + 1}{x^3 + x + 1} \tag{6}$$

is a BCH code of length $n' = 63$ which corrects at least 17 independent errors. Therefore, C is the low-redundancy code in comparison with C'.

3. SHORTENED CODES

When some high degree coefficients are set to zero in a code polynomial, they can be skipped in transmission. Such a shortened code can correct at least the same maximum number of burst or independent errors. Since the same generator polynomial is used, the number of check bits remains the same. The shortened code therefore has more redundancy and more protection per bit.

In many cases, shortened codes are capable of correcting more errors than the original code [5]. The following tables show the lengths and the corresponding maximum bursts of some of the shortened optimum codes [3]. Each of these codes is somewhat longer than the Fire codes with the same maximum burst and the number of check bits. The procedure used to determine the lengths of these shortened codes is the same as that of Kasami's [5].

TABLE I

OPTIMUM BURST - 3 CODES SHORTENED $N(i) \Rightarrow N(b=i)$

g(x)	r	N(3)	N(4)	N(5)	N(6)	N(7)
447	8	63	16	-	-	-
2511	10	255	-	18	-	-
14147	12	1023	-	74	18	-
15517			-	86	-	-
12657			-	-	22	-
11373			-	88	-	-
11711			-	-	21	-
45767	14	4095	-	445	-	-
41367			938	281	93	19
45453			-	280	78	22
62333			-	-	-	23
47613			-	-	-	23
62113			1151	-	-	-
46063			947	-	-	-
53441			1443	-	-	-
51425			-	-	-	24
53625			-	341	-	-

TABLE II

OPTIMUM BURST-4 CODES SHORTENED

g(x)	r	N(4)	N(5)	N(6)
11221	12	511	-	20
24251	13	1023	162	24
101051	15	4095	-	341
107521			809	-
174543			851	-

The generator polynomials are represented by the binary coefficients in octal form. For instance, the 14th line in Table I indicates that g(x) = >46063 = >100,110,000,110,011 = > $x^{14} + x^{11} + x^{10} + x^5 + x^4 + x + 1$. The degree of g(x) is 14. The code generated by g(x) is an optimum burst-3 correcting code of length = 4095. When shortened to 947 bits, it becomes a burst-4 correcting code. The lengths corresponding to bursts of 5, 6, and 7 are too short to be interesting.

A pre-multiplying input circuit is used for the decoding of shortened codes [6]. This circuit is switched out in the low-redundancy mode in which an optimum burst-error correcting code is used.

4. COMBINATION OF SHORTENED CODES AND PRODUCT CODES

This possibility is demonstrated in the following example:

Example 4: Let C_1 be the code generated by

$$g_1(x) = (1 + x)(x^5 + x^4 + x^3 + x^2 + 1) \qquad (7)$$

Then C_1 is a burst-2 correcting code of length $n_1 = 31$.

The code C generated by

$$g(x) = g_1(x)\ (1 + x) \tag{8}$$

corrects all 3-bursts up to the shortened length $N(3) = 2^5 - 5 = 27$. [1]

REFERENCES

1. Chien, R. T., "Shortened Cyclic Codes for Burst Correction," IBM Research Note, NC-349, 1964.

2. Chien, R. T., and Tang, D. T., "Cyclic Product Codes with Variable Redundancy," IBM Research Report, 1964.

3. Elspas, B., and Short, R. A., "A Note on Optimum Burst-Error Correcting Codes," IRE Transactions IT-8, pp. 39-42, January, 1962.

4. Gorog, E., "Some New Classes of Cyclic Codes Used for Burst-Error Correction," IBM Journal of Research and Development, Vol. 7, No. 2, April, 1963.

5. Kasami, T., "Optimum Shortened Cyclic Codes for Burst-Error Correction," IRE Transactions IT-9, pp. 105-109, April, 1963.

6. Peterson, W. W., "Error Correcting Codes," MIT Technology Press, Cambridge, Mass. and Wiley, New York, New York; 1961.

7. Tang, D. T., "Dual Codes as Variable Redundancy Codes," IBM Research Report RC-1122, 1964.

8. Tang, D. T., "On the Implementation of Error Correcting Codes," IBM Research Report, 1964.

APPENDIX E

ON DETECTING ERRORS AFTER CORRECTION*

There seems to be an oversight among the coding theorists that sometimes one is able to detect the presence of errors in the received sequence after error-correction has been performed on it. This has led to the design of systems that possess post-correction error-detecting features. (For example, see reference [1]). The purpose of this note is to demonstrate that although one could incorporate both error-correction and error-detection capabilities into a single code [2] it is never possible to detect errors after error-correction has been made on the received sequence. This is a relatively simple point. However, its importance points up the need of a short note to clarify the situation.

Let us consider any linear code over a finite field. The codewords v satisfy the equation:

$$vH^T = 0 \tag{1}$$

where v is an n-tuple and H is the r by n parity-check matrix defining the code. Denote r = v + e as the received sequence where e is the actual error vector. The syndrome s may be computed through the relation

$$s = rH^T = eH^T \tag{2}$$

We observe that for any fixed error-correction procedure one always subtracts a correction sequence c from the received sequence, and that c is solely determined by the syndromes. Furthermore,

$$cH^T = s \tag{3}$$

From (2) and (3), we may conclude that

$$\begin{aligned}(r - c)H^T &= (v + e - c)H^T \\ &= vH^T + (e - c)H^T \\ &= 0\end{aligned} \tag{4}$$

which means r-c, the received sequence after correction, is always a codeword. Whether this codeword is the same as the sequence transmitted one would never be able to find out short of retransmission.

Cyclic codes form a subclass of linear codes. The error-correction procedures for cyclic codes also perform, directly or indirectly, the same basic function as discussed above. As a result the same conclusion holds.

REFERENCES

1. Bartee, T. C., and Schneider, D. L., "An Electronic Decoder for Bose-Chaudhuri-Hocquenghem Error-Correcting Codes," IEEE Trans on Information Theory, Vol. IT-8, No. 5, pp. S-17-24, September, 1962.

2. Peterson, W. W., "Error Correcting Codes," Wiley, 1961.

APPENDIX F

MODE CONTROL IN COMMUNICATIONS SYSTEMS USING VARIABLE REDUNDANCY CODES

The idea of using variable redundancy codes in a communication system is attractive. Such a system can operate in different modes offering various degrees of error protection. The practicality of this idea, however, depends on whether a good scheme in mode control can be found, and whether efficient code combinations with efficient implementations can be worked out. The purpose of this report is to investigate various ways of having the receiver operate in the correct mode, the mode of the sender. Generally speaking, this would imply some means of informing the receiver of the mode of operation. In some special cases, it is possible to have the receiving end operate in such a way that the knowledge of the mode of operation is unnecessary. Different degrees of error protection in the latter approach are obtained by partitioning the code space in a non-uniform fashion corresponding to two or more different codes.

Considering the price one pays for the multiple-mode operation in terms of hardware and code efficiency, it appears that two modes would probably suffice in most cases. Unless otherwise specified, we shall take a communication system which operates in low-redundancy mode for ordinary messages and in high-redundancy mode for special (command) messages as a standard model. Binary cyclic codes will be used.

When an error in the mode control may result in a chain of erroneously decoded messages, the error protection should be strong. Generally speaking, sufficient flexibility in error control, reasonable complexity in hardware, and efficiency in information transmission are among the basic requirements to all schemes of mode control which will be considered in the following sections.

1. EACH MESSAGE BLOCK CONTAINS A MODE INDICATION

If each message block contains one or several bits exclusively used for the mode indication of the same block which is obtained without decoding, the protection of this mode indication must come from these bits. This apparently results in a sacrifice of many bits if reasonable protection is to be obtained.

A much more practical scheme is for a message block to contain a mode indication of the next message block. A single information bit assigned for this purpose thus gets the same protection as a regular code word. The question which now arises is - What happens to this bit if the message is erroneously decoded?

To answer the above question, let n be the block length, and t be the maximum number of correctible errors in the present mode of operation. Denote the received polynomial by $C_i'(x)$; then

$$C_i'(x) = C_i(x) + E(x), \tag{1}$$

where $C_i(x)$ is the original code polynomial and $E(x)$ is the error occurring during transmission. If $E(x)$ is an uncorrectible error, then $C_i'(x)$ would be erroneously decoded as $C_j(x)$, such that

$$C_i'(x) = C_j(x) + E'(x) \tag{2}$$

where $E'(x)$ is a correctible error. Let the decoding error be $D(x)$, i.e.,

$$D(x) = C_i(x) + C_j(x), \tag{3}$$

then both $C_j(x)$ and $D(x)$ must be code polynomials. From (1), (2), and (3), we have

$$D(x) = E(x) + E'(x). \tag{4}$$

It can be seen that, for most practical cases, $t << n$, and hence $D(x)$ is likely to contain many more zeroes than ones. Since $D(x)$ is a code polynomial, so are all its cyclic permutations. It follows that, even if a message is erroneously decoded, a bit for mode indication still has a very high probability of being correct. One may write the probability of an error in mode indication

$$P_{\text{error ind.}} \simeq P_{\text{code error}} \frac{2(t+1)}{n} \tag{5}$$

as a rough estimate. This is a big improvement over the bit error probability obtained if the bit is used to indicate the mode of operation of the same block. The above analysis also applies to burst-error-correcting codes.

A very desirable feature is obtained when a cyclic product code-pair is used for indicating the mode of the next message block: When the system is to change from low-redundancy mode to high-redundancy mode, an erroneous mode indication does not cause any chain effect. For if C (x) is a code polynomial in high-redundancy mode, it is also a code polynomial in low-redundancy mode. Therefore, the mode indication bit is protected by the low redundancy code and the probability of erroneous mode indication is unchanged. On the other hand, changing from high-redundancy mode to low-redundancy mode, the mode indication already gets more protection from the more powerful code itself. The error propagation in mode indication is thus not serious.

If the probability of erroneous mode indication estimated in (5) is not satisfactory, more bits may be used. This may lead to undesirable complications in decoding, especially when these bits must be spaced to avoid burst errors. An alternative method would be to use these extra bits for a better code. This will result in an overall improvement in the performance of low-redundancy mode.

2. EVERY W'th MESSAGE BLOCK CONTAINS THE MODE INFORMATION

If the mode of operation is not to be changed throughout w blocks, then the problem is essentially the same as that investigated in the last section. If a mode change may occur anywhere within w blocks, every w'th block must convey the information of every such change. This can be done by either giving the exact block numbers at which a change of mode is to take place, or by giving the run length of each mode between mode changes. Either mode can be used for the protection of this information. The maximum allowable number of mode changes within w blocks is determined from the number of information bits available and the maximum number of bits required to indicate a single mode change.

Although the above scheme of mode control may be satisfactory in some special cases, it has some major drawbacks in general:

A. Complicated hardware required to transform the mode information into actual mode control.

B. A maximum delay of w message blocks at the sending end.

When the above problems are not serious, the use of this approach can be considered.

3. USE AN INTERRUPT SEQUENCE FOR THE INDICATION OF MODE CHANGES

In order that a sequence of m bits can be used to interrupt the communication at any time a mode change is to take place, the following conditions must be satisfied:

A. The distance between the interrupt sequence and its shifted versions must be high to assure synchronization.

B. The distance between the m-bit interrupt sequence and any m consecutive bits in the code sequence must be large enough to avoid possible confusion.

Although there are sequences with properties satisfying condition A, it is rather difficult to meet condition B with these sequences alone. To amend such difficulties, an interrupt sequence may consist of two parts: The first part is a sequence satisfying condition B. This could be a sequence of m_1 ones for instance, which can easily be ruled out in the code word part. The second part is a sequence satisfying condition A, such as a maximum length sequence. Thus the first part serves to identify the interrupt sequence, and the second part serves to establish synchronization with error protection.

Let's consider a chain of m_1 ones followed by a maximum length sequence of m_2 bits as follows: (m_2 = 15 in this case)

$$s = 1\ 1\ 1\ 1 \text{-----} 1\ 1\ 1\ 1 \,\vdots\, 0\ 0\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 1 \qquad (6)$$

Since a maximum length sequence of m_2 bits has a binary distance of $\frac{m_2+1}{2}$ between itself and any of its cyclically permuted versions, any one of the cyclically permuted versions has the same property. By placing the longest run of ones at the end of the sequence (a maximum run of four ones in this case), the last few bits of the first part can be considered as part of the maximum length sequence as well. This results in a largest minimum distance between the sequence s and the shifted versions of s. In this case, synchronization is guaranteed in the presence of three independent errors or less within a shift of 15 bits.

The protection against an erroneous identification with the first part of the sequence depends upon the partition of the code space. Usually the message sequence is obtained by packing the m_0-bit (m_0 = 6, for instance, is possible) character representations together. And therefore, by ruling out the m_0-bit character representation of all ones (or m_0-1 ones as well for more protection), confusion from the information part of the code word is well under control. However, it is still possible to have a string of ones in the check bits of some code word. To avoid the possible confusion due to the string of ones in the check bits, m_1 must be considerably larger than r, the total number of check bits, and $r < m_2$.

A possible working scheme is the following: After a string of m_1 ones is recognized (with some allowable errors), the match of the maximum length sequence is attempted over the range between the $(m_1 - m_2 + 1)$'th bit and the $(m_1 + 2m_2)$'th bit, counting from the first bit of the string of ones which match the first part of the sequence.

The recognition of the interrupt sequence takes place prior to the decoding process. It requires a buffer and a logic circuit (for instance, some inversion gates and a threshold device). Part of the buffer storage in an ordinary decoder can be used for this purpose as long as the recognition of the interrupt sequence can be carried out before the information is shifted out of the buffer.

4. USE A SPECIAL CODE WORD FOR THE INDICATION OF MODE CHANGES

The main difference between this method and the use of interrupt sequence is that, with this method, the indication of mode changes is represented by a regular code word, and thus gets its protection from the code. The sequence must start from the beginning of a regular message block and last the whole length of the block.

The code word in which the information consists of all k bits of ones, for instance, is very suitable for the indication of change of mode. Again, a string of ones can be conveniently ruled out in the character representation. Since a number of characters are packed into k bits, the protection of this special code word is much stronger than a regular code word. In the case of product codes [2] where $(x+1) \mid g(x) = g_1(x)\, g_2(x)$, a sequence of n ones could be the special code word. For instance, let

$$g_1(x) = x^6 + x^5 + x^2 + x + 1 \qquad (7)$$

and

$$g(x) = (x^3 + x + 1)\, g_1(x) \tag{8}$$

be the generators of the product code-pair. The code generated by g(x) is a burst-3 correcting code. The code word which contains all 63 bits of ones is protected against four independent errors if the all-one sequence in a 6-bit character representation is ruled out.

The recognition of the special code word can be easily carried out with the similar circuit containing a threshold device and perhaps some inverters as described before. Here we do not have the problem of synchronization.

5. NON-UNIFORM PARTITION OF THE BINARY SPACE

The whole encoding and decoding process can be considered as a classification process. A number of disjoint partitions are first obtained. Each partition is then mapped into a code word contained in the partition. Usually these partitions are uniform in size and similar in their relationship to the code words they map into. However, the partitions can be non-uniform and different protections thus obtained. The trouble with this approach is that it is often difficult to find a class of partitions with reasonably simple implementation. However, it is attractive in the fact that in general no special bits, sequences, or code words are exclusively used for mode indication. The information is conveyed in a more subtle manner.

Two examples are shown below:

A. Let $g_1(x)$ be the generator polynomial of a t independent error (or a b-burst) error detecting code, and let $g(x) = g_1(x)\ g_2(x)$ be the generator polynomial of a t-error (or b-burst) correcting code. A low-redundancy code polynomial $C_\ell(x)$ satisfies the following equations:

$$C_\ell(x) \equiv 0 \qquad \text{mod } g_1(x), \tag{9}$$

and

$$C_\ell(x) \not\equiv 0 \qquad \text{mod } g(x). \tag{10}$$

A high-redundancy code polynomial $C_h(x)$ satisfies

$$C_h(x) \equiv 0 \qquad \text{mod } g(x). \tag{11}$$

In the process of decoding, the residues of a received sequence $C(x)$ is calculated mod $g_1(x)$ and mod $g(x)$. If

$$C(x) \equiv 0 \qquad \text{mod } g_1(x), \tag{12}$$

and

$$C(x) \not\equiv 0 \qquad \text{mod } g(x), \tag{13}$$

it is recognized as a low-redundancy code polynomial.

If

$$C(x) \not\equiv 0 \qquad \text{mod } g_1(x), \tag{14}$$

the error correction is carried out according to the code generated by $g(x)$.

Figure F-1 shows a decoding circuit of an above scheme. Using the building blocks [3] of $g_1(x)$ and $g_2(x)$, residue conditions of $C(x)$ mod $g_1(x)$ and mod $g(x)$ can be checked at the same time. Note that since

$$t = t_1 + t_2 \tag{15}$$

and

$$s = s_1 + t_2 \tag{16}$$

we have

$$m + t + s = m + s_1 + t_1 = 0 \tag{17}$$

It follows that the contents of G_1-register behaves as if it were connected as a division circuit.

Switch M is controlled by the zero detect signal of the G_1-register right after the complete code word of n bits has entered the circuit. Switch M will stay at position 1 if the detecting circuit finds the contents of G_1-register to be all zeroes and will stay at position 2 if the detection of all zero fails.

The main disadvantage of this approach is that the low-redundancy codes do not get any protection. It therefore can only be used when this situation is acceptable. The high-redundancy codes get the protection against t errors.

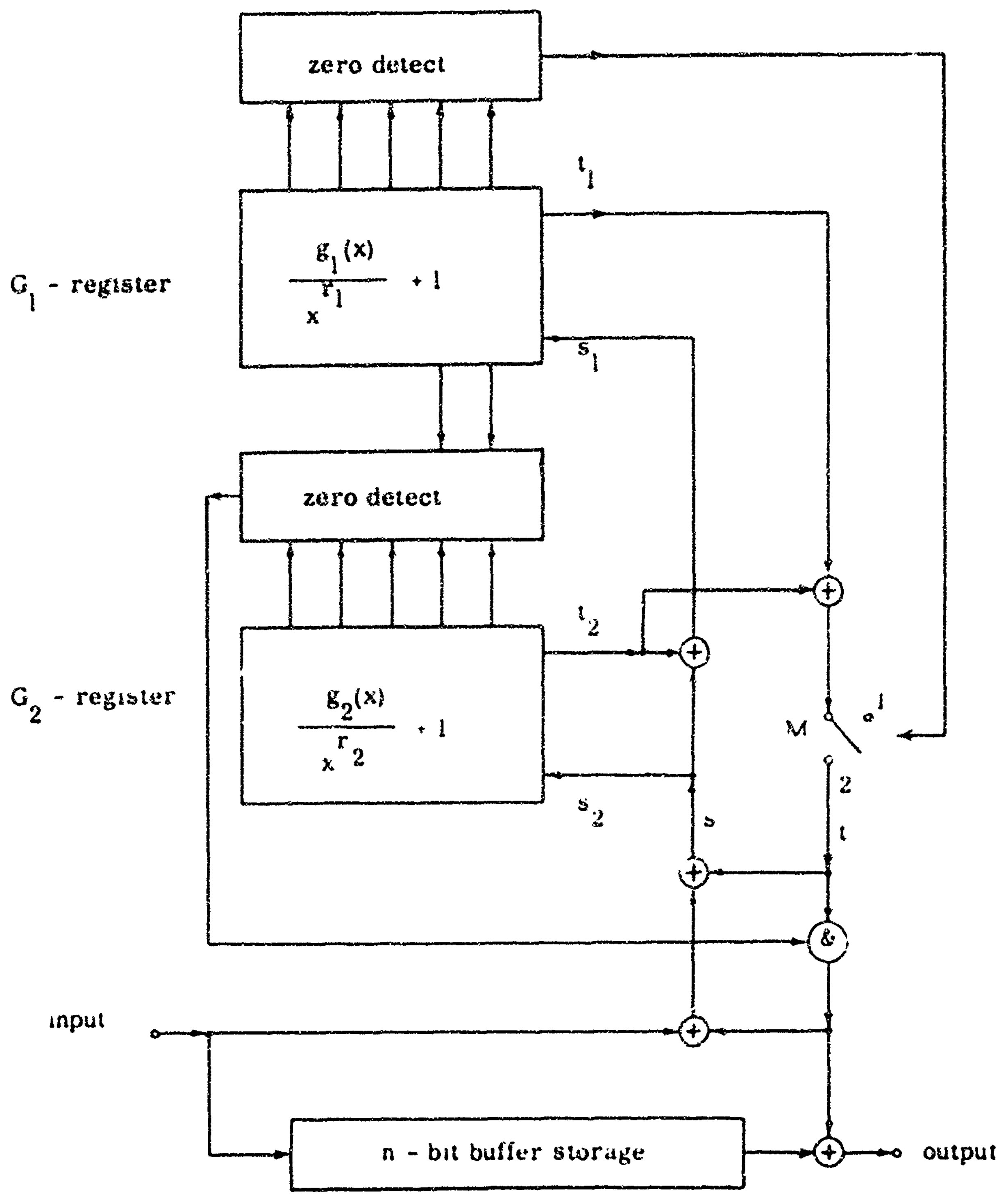

zero detect
t_1
G_1 - register
$\frac{g_1(x)}{x^{r_1}} + 1$
s_1
zero detect
t_2
G_2 - register
$\frac{g_2(x)}{x^{r_2}} + 1$
s_2
s
M
1
2
t
&
input
n - bit buffer storage
output

Figure F-1

B. Another way to partition the n-dimensional binary space is as follows: Let $g_1(x)$ and $g(x) = g_1(x)\, g_2(x)$ be the generators of a product code pair. The high-redundancy code polynomials consist of multiples of $g(x)$ such that the first bit (coefficient of x^{n-1}) is a "1". The low-redundancy code polynomials consist of multiples of $g_1(x)$ such that, when decoded according to $g(x)$, the first bit is a "0".

At the receiving end, syndromes with respect to both $g_1(x)$ and $g(x)$ are generated. A logic circuit is used to sense the first bit of the message based on the code structure of $g(x)$. This signal then switches the decoding circuit to the proper mode of operation.

Since one bit is used for mode indication, this approach is somewhat similar to that of Section 1. It is also similar to the approach discussed in Section 4, especially when several modes are present and a number of special code words are necessary.

Figure F-2 shows a decoding circuit for the product code pair:

$$g_1(x) = x^4 + x + 1, \tag{18}$$

and

$$g(x) = (x^4 + x + 1)\ (x^2 + x + 1), \tag{19}$$

where $g_1(x)$ corresponds to mode 1 and $g(x)$ corresponds to mode 2.

After the n-bit ($n = 15$ here) received message enters the circuit, the M-set output is obtained by triggering a sensing pulse which does not cause the register contents to shift. Switch M is originally at position 2 and will stay there if the M-set output is one (mode 2); it will switch to position 1 if the M-set output is zero (mode 1). The main disadvantage of this approach is that the low-redundancy code is not separable. Therefore, recovering the original code word may not be sufficient.

6. CONCLUSION

Five approaches of mode control have been investigated in this report. Each one may be used in some special cases to meet the system's requirements, such as the reliability of mode indication, complexity of hardware to instrument mode control, etc.

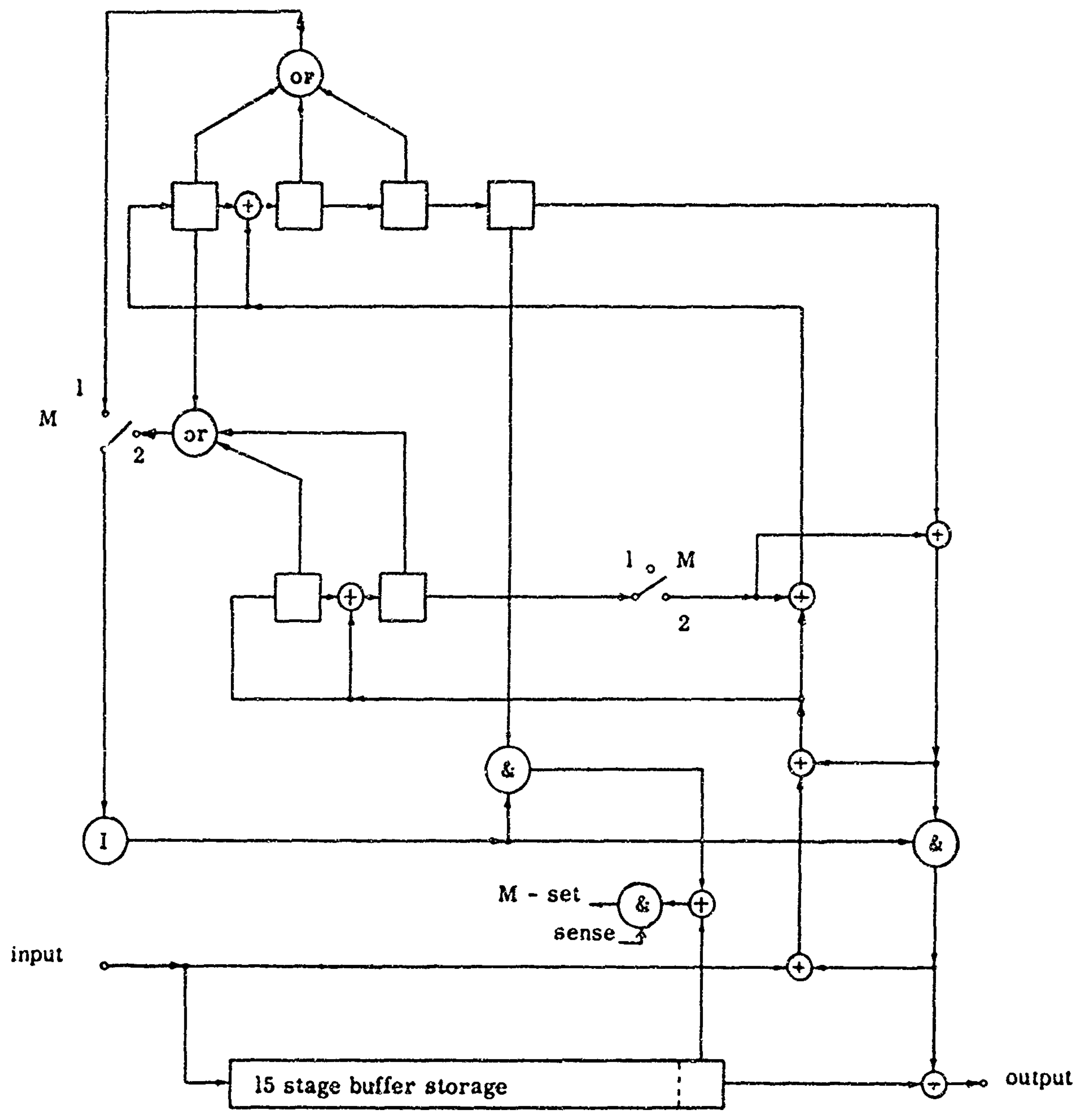


Figure F-2

When the change of mode occurs very often, the approach in which each block contains the mode indication (Section 1) seems appropriate. On the other hand, if the change of mode does not occur very often, the use of a special code word (Section 4) or the use of an interrupt sequence (Section 3) seems better. The use of interrupt sequence may be preferred when the code length is very long. The approach of partitioning the n-dimensional binary space in a non-uniform fashion (Section 5) may be considered when the one-to-one mapping between a message and a code polynomial does not present any problem at both sending and receiving ends.

REFERENCES

1. Chien, R. T., and Tang, D. T., "On Detecting Errors After Correction," IBM Research Note NC-333, December, 1963.

2. Chien, R. T., and Tang, D. T., "Cyclic Product Codes with Variable Redundancy," IBM Research Report, 1964.

3. Tang, D. T., "On the Implementation of Error-Correcting Codes," IBM Research Report, 1964.

4. Tang, D. T., "Dual Codes as Variable Redundancy Codes," IBM Research Report, RC-1122, January, 1964.

5. Tang, D. T., "On Variable-Length Codes as Variable-Redundancy Codes," IBM Research Note, 1964.

APPENDIX G

BURST-ERROR CYCLIC CODES GENERATED FROM CYCLIC GROUPS

Elspas and Short [1] has searched for optimum codes for burst errors. His approach becomes very difficult to handle even by computers for $b > 5$. As Fire Codes are relatively inefficient for short length, there exists a need for knowing more codes in the area where $n \leq 4095$ and $r \leq 30$ for practical applications.

In this note we report a class of codes obtained through a limited search with the aid of some theoretical guide lines. These codes are obtained from the class where the error patterns are all relatively prime to the generator polynomial. A method utilizing the structure of cyclic groups makes the search very efficient and not time-consuming. The following lemmas provide the necessary theoretical foundation of the procedure.

LEMMA 1

If $(\phi(q_1), \phi(q_2)) = 1$, then $(q_1, q_2) = 1$.

Proof: Suppose $q_1 = q'_1 q_0$, $q_2 = q'_2 q_0$. Let the periods of q_0, q_1, q_2, q'_1, and q'_2 be n_0, n_1, n_2, n'_1 and n'_2 respectively. Then, n_1 is divisible by L.C.M. (n_0, n'_1) and n_2 is divisible by L.C.M. (n_0, n'_2). Hence n_0 divides both n_1 and n_2. Furthermore $n_1 | \phi(q_1)$ and $n_2 | \phi(q_2)$. Consequently, $\phi(q_1)$ and $\phi(q_2)$ have a common factor n_0. Hence if $(\phi(q_1), \phi(q_2)) = 1$, $(q_1, q_2) = 1$.

LEMMA 2

Let $q = q_1 q_2$; then Q is cyclic if and only if

(1) Q_1 is cyclic

Q_2 is cyclic

(2) $(\phi(q_1), \phi(q_2) = 1$,

where Q, Q_1, Q_2 are multiplicative groups modulo q, q_1, and q_2 respectively.

Proof: Suppose Q_1 is cyclic with primitive root α

Q_2 is cyclic with primitive root β

By Lemma 1, $(\phi(q_1), \phi(q_2)) = 1$ implies $(q_1, q_2) = 1$. Then, there exists

$$\gamma \equiv \alpha \qquad (q_1)$$

$$\gamma \equiv \beta \qquad (q_2)$$

and

$$\gamma \equiv Z_1\alpha + Z_2\beta \qquad (q_1 q_2)$$

where

$$Z_1 \equiv 0 \qquad (q_2)$$

$$\equiv 1 \qquad (q_1)$$

$$Z_2 \equiv 0 \qquad (q_1)$$

$$\equiv 1 \qquad (q_2)$$

Furthermore,

$$\gamma^{\phi(q_1)} \equiv 1 \qquad (q_1)$$

$$\gamma^{\phi(q_2)} \equiv 1 \qquad (q_2)$$

and

$$\gamma^{\delta} \equiv 1 \ (q_1) \text{ implies } \phi(q_1) \mid \delta$$

$$\gamma^{\delta} \equiv 1 \ (q_2) \text{ implies } \phi(q_2) \mid \delta$$

Hence

$$\gamma^{\delta} \equiv 1 \ (q_1 q_2) \text{ implies L.C.M. } (\phi(q_1), \phi(q_2)) \mid \delta$$

But

$$(\phi(q_1), \phi(q_2)) = 1.$$

This means

$$\delta = \phi(q_1) \cdot \phi(q_2)$$

and Q is cyclic with primitive root γ. Conversely, assuming Q is cyclic with primitive root γ,

suppose

$$\gamma^{\delta} \equiv 1 \ (q_1);$$

then $\delta \geq \phi(q_1)$, for $\gamma^{\delta} \equiv 1$ implies $\alpha^{\delta\phi(q_2)} \equiv 1 \ (q_2)$, which in turn implies $\gamma^{\delta\phi(q_2)} \equiv 1 \ (q_1 q_2)$.

Hence, γ is a primitive root of Q_1. Similarly, γ is a primitive root of Q_2.

Suppose $(\phi(q_1), \phi(q_2)) = d$;

then

$$L = \text{L.C.M.} \ (\phi(q_1), \phi(q_2)) = \frac{\phi(q_1)\,\phi(q_2)}{d}$$

But

$\phi(q_1)\,\phi(q_2) \mid L$. Hence, they are equal and $d = 1$.

LEMMA 3

Q is cyclic if and only if

$$q = p_1 p_2 p_3 \text{--} p_m \cdot p_i^m$$

where

$$(\phi(p_i), \phi(p_j)) = 1.$$

Proof: For $m = 0$ this assertion follows directly from Lemma 2 by induction. It is known that if the period of x modulo p_i is n_i then the period of x modulo p_i^m is $2^k n_i$ where

$$2^k \leq m < 2^{k+1}$$

For instance, see [2]. Hence the Lemma follows.

LEMMA 4

Let Q be cyclic and n be the period of x modulo q(x). Then, the set of polynomials $(B_i(x))$ are in distinct cosets if and only if

$$(B_i(x))^n$$

are distinct.

Proof: Let α be a primitive root of Q, then $x = \frac{m}{n}$ where m is the order of Q. The group Q may be expanded into cosets with coset leaders

$$1,\ \alpha,\ \alpha^2,\ \alpha^3, --- \alpha^{\left(\frac{m}{n}\right)-1}$$

Let $B_i(x) = \alpha^{i_1 \frac{m}{n} + i_2}$;

then

$$\left(\alpha^{i_1 \frac{m}{n} + i_2}\right)^n = \left(\alpha^{j_1 \frac{m}{n} + j_2}\right)^n$$

if and only if

$$i_2 = j_2$$

Table I shows some burst-error-correcting codes with $b \geq 5$. The codes were obtained by checking Lemma 4 with a computer program. All these generator polynomials consist of two irreducible factors with a period less than 4000. More codes can be found in a similar manner which cover a larger range of degrees, periods and number of factors of the generator polynomial.

The polynomials in Table I are represented by its binary coefficient in octal form. For instance, $4115 \rightarrow 100,001,001,101 \rightarrow x^{11} + x^6 + x^3 + x^2 + 1$.

TABLE I

r	q(x)	b	n
11	4115 7225 6117	5	279
13	31475 23201 22133 23467 32123 21465 37215 35217 25207	5	1143
14	61425 67423 43407 75257	5 6	765 153
15	145501 130021 155511 113557	5	657
16	255611 223667 241063 361143 233677 273637 210301 355177 265047 301745	5	3069

TABLE I (Continued)

r	q(x)	b	n
16	205027	5	3069
	341163		
	335771		
	320231		
	343707	6	99
	313715	6	231
18	1510155	6	4095
	1313515	8	187
	1643427		187
22	25256525	10	143

REFERENCES

1. Elspas, B., and Short, R. A., "A Note on Optimum Burst-Error Correcting Codes," IRE Transactions, IT-8, pp. 39-42, January, 1942.

2. Elspas, B., "The Theory of Autonomous Linear Sequential Networks," IRE Transactions, CT-6, pp. 45-60, March, 1959.